Unconditional Lower Bounds in Complexity Theory

Igor Carboni Oliveira

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2015

ABSTRACT

Unconditional Lower Bounds in Complexity Theory

Igor Carboni Oliveira

This work investigates the hardness of solving natural computational problems according to different complexity measures. Our results and techniques span several areas in theoretical computer science and discrete mathematics. They have in common the following aspects: (*i*) the results are *unconditional*, i.e., they rely on no unproven hardness assumption from complexity theory; (*ii*) the corresponding lower bounds are *essentially optimal*. Among our contributions, we highlight the following results.

- *Constraint Satisfaction Problems and Monotone Complexity.* We introduce a natural formulation of the satisfiability problem as a monotone function, and prove a near-optimal $2^{\Omega(n/\log n)}$ lower bound on the size of monotone formulas solving k-SAT on n-variable instances (for a large enough $k \in \mathbb{N}$). More generally, we investigate constraint satisfaction problems according to the geometry of their constraints, i.e., as a function of the hypergraph describing which variables appear in each constraint. Our results show in a certain technical sense that the monotone circuit depth complexity of the satisfiability problem is polynomially related to the tree-width of the corresponding graphs.

- *Interactive Protocols and Communication Complexity.* We investigate interactive compression protocols, a hybrid model between computational complexity and communication complexity. We prove that the communication complexity of the Majority function on n-bit inputs with respect to Boolean circuits of size s and depth d extended with modulo p gates is precisely $n/\log^{\Theta(d)} s$, where p is a fixed prime number, and $d \in \mathbb{N}$. Further, we establish a strong round-separation theorem for bounded-depth

circuits, showing that $(r+1)$-round protocols can be substantially more efficient than r-round protocols, for every $r \in \mathbb{N}$.

- *Negations in Computational Learning Theory.* We study the learnability of circuits containing a given number of negation gates, a measure that interpolates between monotone functions, and the class of all functions. Let $\mathcal{C}_n^t$ be the class of Boolean functions on n input variables that can be computed by Boolean circuits with at most t negations. We prove that any algorithm that learns every $f \in \mathcal{C}_n^t$ with membership queries according to the uniform distribution to accuracy ε has query complexity $2^{\Omega(2^t\sqrt{n}/\varepsilon)}$ (for a large range of these parameters). Moreover, we give an algorithm that learns $\mathcal{C}_n^t$ from random examples only, and with a running time that essentially matches this information-theoretic lower bound.

- *Negations in Theory of Cryptography.* We investigate the power of negation gates in cryptography and related areas, and prove that many basic cryptographic primitives require essentially the maximum number of negations among all Boolean functions. In other words, cryptography is highly non-monotone. Our results rely on a variety of techniques, and give near-optimal lower bounds for pseudorandom functions, error-correcting codes, hardcore predicates, randomness extractors, and small-bias generators.

- *Algorithms versus Circuit Lower Bounds.* We strengthen a few connections between algorithms and circuit lower bounds. We show that the design of faster algorithms in some widely investigated learning models would imply new unconditional lower bounds in complexity theory. In addition, we prove that the existence of non-trivial satisfiability algorithms for certain classes of Boolean circuits of depth $d+2$ leads to lower bounds for the corresponding class of circuits of depth d. These results show that either there are no faster algorithms for some computational tasks, or certain circuit lower bounds hold.

Table of Contents

Acknowledgments

I would like to thank Tal Malkin, Rocco Servedio, Mihalis Yannakakis, and Clifford Stein, for supporting my admission as a graduate student in the Theory of Computation Group. I am grateful to Columbia University for the fantastic environment that allowed me to carry out my research.

I thank the members of my thesis committee, Andrej Bogdanov, Xi Chen, Tal Malkin, Rahul Santhanam, and Rocco Servedio, for their time and valuable feedback.

Tal and Rocco, thank you for generously sharing so much time with me during these five intense years. I am eternally grateful to all you have taught me, for the freedom that allowed me to investigate the questions that excite me, for the conversations and enlightening advice, and for your friendship. It was a privilege to have both of you advising me during this journey as a doctoral student.

I wish to also thank Walter Carnielli, Orlando Lee, and Arnaldo Moura, for guiding me in my first steps in research with patience and enthusiasm. I thank Cid Carvalho de Souza for introducing me to computational complexity theory through his lectures on algorithms and complexity.

Clément, Dimitris, Eva, and Fernando, it was a pleasure to share the office with you. Xi Chen, it was great to share the fifth floor with you. Thank you for making my days less lonely in New York.

I would like to express my thanks to Yoshiharu Kohayakawa for hosting me at University of São Paulo, where I was fortunate to overlap with Andrea, Hiệp, Marcelo, and Chandu. I thank you all for the very enjoyable time in Brazil.

I also thank Alon Rosen for inviting me for a stay in Israel, and Ilan, Silas, Siyao, and Tal for their company in beautiful Herzliya during that period.

My special thanks to Rahul Santhanam for hosting me at University of Edinburgh, during what was a very enjoyable collaboration, and for the innumerable forms of support since then.

I would like to thank Andrej Bogdanov and Siyao Guo for inviting me for a short but warm visit to Hong Kong, and for the many walks and discussions throughout the city.

I wish to thank Rahul Santhanam, Jan Krajíček, and Olaf Beyersdorff for inviting me to the Dagstuhl Workshop "Optimal Algorithms and Proofs," and Valentine Kabanets and Ryan Williams for the invitation to the Simons Workshop "Connections Between Algorithm Design and Complexity Theory." Many thanks to Ben, Rafael, and Susanna for making going to workshops and conferences a lot funnier.

I had the luck to work with several researchers while in graduate school. I thank Adam Klivans, Pravesh Kothari, Bhalchandra Thatte, Marcelo Gauy, Hiệp Hàn, Siyao Guo, Tal Malkin, Alon Rosen, Rahul Santhanam, Eric Blais, Clément Canonne, Rocco Servedio, Li-Yang Tan, and Xi Chen for all I have learned from them during these collaborations. I also thank Andy Drucker, Ben Rossman, and Sasha Golovnev for stimulating discussions that influenced the results in this thesis.

Sincere thanks to my friends from Brazil, including Felipe, Allan, Carolina, Campello, João Tiago, Limão, Digão, and Douglas.

I thank the hospitality of the staff members of Instituto de Física Teórica (UNESP) and Rotch Library (MIT), where I spent a reasonable amount of time thinking about the results of this thesis.

Finally, and above all, I thank Carlos, Helena, Iana, and Nayara, for their unconditional love and support. There are no words to express your importance in my life.

Bibliographic Note

Chapter 3 is the result of individual work. A research paper containing the contributions of this chapter is in preparation.

Chapter 4 is based on the paper "Majority is incompressible by $\mathsf{AC}^0[p]$ circuits," which is joint work with Rahul Santhanam, and will appear in the Proceedings of the 30th Conference on Computational Complexity (CCC 2015).

Chapter 5 is based on the paper "Learning circuits with few negations," which is joint work with Eric Blais, Clément Canonne, Rocco Servedio, and Li-Yang Tan, and will appear in the Proceedings of the 19th International Workshop on Randomization and Computation (RANDOM 2015).

Chapter 6 is based on the paper "The power of negations in cryptography," which is joint work with Siyao Guo, Tal Malkin, and Alon Rosen, and will appear in the Proceedings of the 12th Theory of Cryptography Conference (TCC 2015).

Chapter 7 is based on the paper "Constructing hard functions using learning algorithms," which is joint work with Adam Klivans and Pravesh Kothari, and appeared in the Proceedings of the 28th Conference on Computational Complexity (CCC 2013).

Chapter 8 is the result of individual work. A preliminary version of the results of this chapter appeared in the technical report ECCC:TR13-117 entitled "Algorithms versus Circuit lower bounds."

Chapter 1

Introduction

1.1 Complexity Theory

Complexity Theory is a relatively new discipline situated at the intersection of Theoretical Computer Science and Mathematics. Following Razborov [Chapter 8, 169], we can describe the main problems investigated in the field roughly as follows. Assume that there is a "computational task" T_n that we want to complete. For instance, T_n may be one of the following:

- Learn a class of functions $\mathcal{C}_n$ (cf. Kearns and Vazirani [117]);
- Compute a Boolean function $f_n \colon \{0,1\}^n \to \{0,1\}$ (cf. Jukna [109]);
- Prove a mathematical statement ϕ_n (cf. Krajícek [126]);
- Solve a communication problem π_n (cf. Kushilevitz and Nisan [128]); etc.

After fixing the task, we consider a class of structures $\mathcal{P}_n$ ("solutions") that solve T_n. For example, depending on the task, $\mathcal{P}_n$ may be a set of algorithms, Boolean circuits, propositional proofs, interactive protocols, etc. Given any object $\mathcal{O} \in \mathcal{P}_n$ that solves T_n, there is an associated value that measures the "complexity" of this solution. Formally, given $\mathcal{P}_n$, we assume the existence of some natural complexity measure

$$\alpha_n \colon \mathcal{P}_n \to \mathbb{N}$$

that assigns to each solution its corresponding complexity. For instance, α_n could be a measure of the number of queries of the solution, circuit size, proof length, communication cost, amount of randomness, etc.

Given a task T_n, a set of admissible solutions $\mathcal{P}_n$, and a complexity measure α_n, we would like to estimate

$$\mu(n) \stackrel{\text{def}}{=} \min_{\mathcal{O} \in \mathcal{P}_n} \alpha_n(\mathcal{O}).$$

In other words, $\mu(n)$ is the complexity of the best solution for our original task T_n.

It may be very hard to determine $\mu(n)$ exactly, or one may want to avoid unnecessary technical details. For these reasons, we usually consider a sequence $\mathcal{T} = \{T_n\}_{n \in \mathbb{N}}$ of tasks, and investigate the *asymptotics* of $\mu(n)$ as $n \to \infty$, where n is a parameter related to the size of the input instances.

This is of course a very high-level description of the problems investigated in complexity theory, since the definitions given above can be used to capture most optimization problems. What gives the field its distinctive flavor is the class of computational models that are used in the definition of $\mathcal{P}_n$, and the associated complexity measures α_n. Due to its importance, and in order to present a concrete example, we discuss in the next two paragraphs *computational* complexity theory.

A widely investigated model of computation is the *Turing Machine*, which is one of several equivalent frameworks that can be used to define algorithms (see e.g. Sipser [175]). In this case, the task $\mathcal{T}$ is simply a sequence of Boolean functions $\{f_n\}_{n \in \mathbb{N}}$ that encodes a computational problem, $\mathcal{P}$ is the set of Turing machines that correctly solve this problem, and $\alpha_n(M)$ denotes the maximum time complexity of a Turing machine M on inputs of size n. (The definitions given above have to be adapted slightly, since this is a *uniform* model of computation, i.e., $\mathcal{P} = \mathcal{P}_n$ for every n.)

The seminal work of Hartmanis and Stearns [94], published fifty years ago, established that with more resources one can solve more computational problems. There exist therefore *hierarchies* of computational problems defined according to natural complexity measures, such as computational time and space. Similar hierarchies are known to hold for many other frameworks investigated in complexity theory, such as *proof* complexity, *communication* complexity, and *circuit* complexity. The existence of such hierarchies implies that

understanding the complexity of the computational tasks in these models is a non-trivial problem.

The most basic questions in complexity theory are determining the complexity of *natural* tasks, and understanding the relation between *different* complexity measures. Since there are numerous connections between the many subareas of complexity theory, it makes sense to group them into a single field of investigation.

1.2 Different flavors of lower bounds

We can distinguish research in complexity lower bounds roughly as follows: "conditional" complexity theory, and "unconditional" complexity theory. In a *conditional* result, one assumes a particular hypothesis, such as $\mathsf{P} \neq \mathsf{NP}$, and tries to explain the hardness of other computational tasks. A substantial number of results in complexity theory are conditional, i.e., are based on a hardness assumption. On the other hand, progress on *unconditional* results has been much slower, since proving limitations for concrete models without extra assumptions requires a deeper understanding of the model. Nevertheless, progress on conditional results can provide the basis for unconditional lower bounds, and these two research directions complement each other. (We will see an example of this phenomenon in Chapter 3.)

We focus on *unconditional* complexity theory in this work. Among unconditional lower bounds, we would like to highlight two types of results: *information-theoretic* and *computational* lower bounds. While this distinction is not always possible, most results presented in this work fit nicely under this classification.

In an information-theoretic lower bound a particular task cannot be completed within certain resources due to an information bottleneck. Put another way, in these lower bounds the objects solving a given task do not have complete information about their input. In particular, there may be no efficient solution to the problem in hand simply because any solution requires the inspection of a large amount of data. For instance, we will investigate learning algorithms in Chapter 5, where one can prove that in order to learn some classes of Boolean functions there is a minimum amount of information that needs to be obtained

about the unknown function.

In a computational lower bound the objects solving the task have complete information about the problem under consideration. Consequently, in order to prove that a task of this form cannot be solved within certain resources, one has to exploit some structural limitation of the class of objects solving the task. In other words, all required information is available, but there is a computational bottleneck that prevents the data from being processed efficiently. We will prove lower bounds of this form in Chapters 3 and 4, for instance.

Most lower bounds discussed in this thesis concern the inherent complexity of performing certain *computations*. In order to study the complexity of computations, we need to fix a convenient computational model.

1.3 The Boolean circuit model

The Boolean circuit is a central model in computational complexity theory. It has well-known connections to several other models, including Turing machines, propositional proof systems, logical theories, and communication protocols (see for instance the textbooks [109, 117, 126, 128] mentioned above). All results in this thesis are connected to Boolean circuits.

For convenience of the reader, we review some definitions. A circuit is a very basic computational model, where a function is computed through a sequence of simple operations. Given functions $f, g\colon \{0,1\}^n \to \{0,1\}$, we define new Boolean functions obtained from them in the natural way via the Boolean operations $f(x) \wedge g(x)$, $f(x) \vee g(x)$, and $\neg f(x)$. The new function is called the *conjunction*, *disjunction*, or *negation* of the involved function(s), respectively. A *Boolean circuit* $C_n = (g_1, \ldots, g_n, g_{n+1}, \ldots, g_s)$ is a sequence of functions, where each $g_i\colon \{0,1\}^n \to \{0,1\}$ is the projection function $g_i(x) = x_i$ for $1 \leq i \leq n$, or is obtained from previous functions via a conjunction, disjunction, or negation, for $n+1 \leq i \leq s$. The Boolean function computed by C_n is $g_s\colon \{0,1\}^n \to \{0,1\}$, the last function in this sequence.

A circuit C_n can also be viewed as a *directed acyclic graph*, with input nodes $x_1, \ldots, x_n$,

internal nodes (gates) labeled by $\wedge$, $\vee$, or $\neg$, and wires connecting each internal node to its children. There is a specially designated *output gate*, and computation is defined in the natural way. It is easy to see that the two definitions are equivalent. The *depth* of a circuit C_n is the length of the longest path from the output gate to an input node, and will be denoted by $\mathsf{depth}(C_n)$. The *size* of the circuit is measured by its number of internal nodes, and will be denoted by $\mathsf{size}(C_n)$.

We say that a circuit is a *formula* if every internal node has fan-out 1. In other words, the output of an internal gate cannot feed multiple gates. Further, a circuit is *monotone* if it contains no negation gates. It is convenient sometimes to allow Boolean circuits to have arbitrary fan-in, i.e., each gate of the circuit can get as input the output of several gates. This is particularly important when we restrict attention to circuits of small depth, since otherwise the output gate cannot depend on all input variables.

It is possible to show by a simple probabilistic argument that a random Boolean function $f_n\colon \{0,1\}^n \to \{0,1\}$ almost surely requires fan-in two circuits of size $\Omega(2^n/n)$ and depth $\Omega(n)$ (Shannon [174]). Put another way, most Boolean functions are hard to compute. Although there has been an intensive effort to understand the complexity of natural Boolean functions, we have not yet succeeded in proving that an explicit Boolean function requires fan-in two circuits of size $10n$. Similarly, we have no proof that an explicit function requires circuits of depth $10 \log n$. We refer the reader to Jukna [109] for an accurate description of the strongest known explicit lower bounds against general Boolean circuits.

Although progress on unconditional lower bounds with respect to general circuits has been almost nonexistent, strong results are known under natural restrictions on Boolean circuits. Two widely investigated restricted classes of Boolean circuits are *bounded-depth* circuits of arbitrary fan-in (see e.g. Håstad [95, 96]), and *monotone* circuits of fan-in two (cf. Razborov [155, 156] and Rossman [161]).

1.4 Main contributions and outline of this thesis

We give a brief description of our contributions in the next subsections. We stress that all results discussed below are unconditional, i.e., they require no hardness assumption.

1.4.1 Part I: Circuit Lower Bounds.

In this part of the thesis, we prove new lower bounds for bounded-depth circuits and monotone circuits.

Chapter 3: On the monotone complexity of the satisfiability problem. We introduce a natural formulation of the satisfiability problem as a *monotone* function over $\binom{n}{k} \cdot 2^k$ input bits, and prove a near-optimal $2^{\Omega(n/\log n)}$ lower bound on the size of monotone formulas computing k-SAT on n-variable instances (for a large enough $k \in \mathbb{N}$). The same lower bound holds when the problem is restricted to instances with a linear number of clauses. This result relies on a lower bound recently obtained by Göös and Pitassi [87].

More generally, we describe a framework to study the monotone circuit depth complexity of constraint satisfaction problems (CSP) based on the geometry of their constraints, i.e., as a function of the hypergraph describing which variables appear in each constraint. We establish, unconditionally, that the monotone depth complexity of the satisfiability problem is connected to the tree-width of the corresponding graphs. Roughly speaking, for any graph G, we consider a related hypergraph $\mathcal{H}$, and prove that a certain satisfiability problem with geometry $\mathcal{H}$ admits shallow monotone circuits if and only if G is reasonably close to a tree.

Our characterization is similar in spirit to a conditional result of Grohe [90] from parameterized complexity theory, but does not require the CSPs to have unbounded domain size. To our knowledge, this is the first hardness result of this form that holds with respect to bounded-size domain CSPs, even among conditional results.

Chapter 4: Majority is incompressible by $\mathsf{AC}^0[p]$ circuits. We consider $\mathcal{C}$-compression games, a hybrid model between computational and communication complexity. A $\mathcal{C}$-compression game for a function $f\colon \{0,1\}^n \to \{0,1\}$ is a two-party communication game, where the first party Alice knows the entire input x but is restricted to use strategies computed by $\mathcal{C}$-circuits, while the second party Bob initially has no information about the input, but is computationally unbounded. The parties implement an *interactive* communication protocol to decide the value of $f(x)$, and the *communication cost* of the protocol is the maximum number of bits sent by Alice as a function of $n = |x|$.

We show that any $\mathsf{AC}^0_d[p]$-compression protocol to compute $\mathsf{Majority}_n$ requires commu-

nication $n/(\log n)^{2d+O(1)}$, where p is prime, and $\mathsf{AC}^0_d[p]$ denotes polynomial size unbounded fan-in depth-d Boolean circuits extended with modulo p gates. This bound is essentially optimal, and settles a question of Chattopadhyay and Santhanam [45]. This result has a number of consequences, and yields a tight lower bound on the total fan-in of oracle gates in constant-depth oracle circuits computing $\mathsf{Majority}_n$.

We define *multiparty* compression games, where Alice interacts in parallel with a polynomial number of players that are not allowed to communicate with each other, and communication cost is defined as the sum of the lengths of the longest messages sent by Alice during each round. In this setting, we prove that the randomized r-round $\mathsf{AC}^0[p]$-compression cost of $\mathsf{Majority}_n$ is $n^{\Theta(1/r)}$. This result implies almost tight lower bounds on the maximum individual fan-in of oracle gates in certain restricted bounded-depth oracle circuits computing $\mathsf{Majority}_n$. Stronger lower bounds for functions in NP would separate NP from NC^1.

Finally, we consider the round separation question for two-party AC^0-compression games, and significantly improve known separations between r-round and $(r+1)$-round protocols, for any constant r.

1.4.2 Part II: Negations in Learning Theory and Cryptography.

The results from Chapter 3 show that certain aspects of the complexity of monotone circuits are well-understood. Extending these results to circuits with arbitrarily many negations remains a major challenge in complexity theory, also known as the NP versus NC^1 problem.

In this part of the thesis, we investigate non-monotone circuits in Computational Learning Theory and Theory of Cryptography.

Chapter 5: Learning circuits with negations. In this chapter we study the structure of Boolean functions in terms of the minimum number of negations in any circuit computing them, a complexity measure that interpolates between monotone functions and the class of all functions. We study this generalization of monotonicity from the vantage point of learning theory, giving near-matching upper and lower bounds on the uniform-distribution learnability of circuits in terms of the number of negations they contain. Our results indicate

a smooth transition from the complexity of learning monotone functions, to the complexity of learning all Boolean functions.

More precisely, let $\mathcal{C}_n^t$ be the class of Boolean functions on n input variables that can be computed by Boolean circuits with at most t negations. We prove that any algorithm that learns every $f \in \mathcal{C}_n^t$ with membership queries according to the uniform distribution to accuracy ε has query complexity $2^{\Omega(2^t\sqrt{n}/\varepsilon)}$ (for a large range of these parameters). Further, we give an algorithm that learns $\mathcal{C}_n^t$ from random examples only, and with a running time that essentially matches this lower bound.

Our upper bounds are based on a new structural characterization of negation-limited circuits that extends a classical result of Markov [132]. Our lower bounds, which employ Fourier-analytic tools from hardness amplification, give new results even for circuits with no negations (i.e. monotone functions).

Chapter 6: The power of negations in Cryptography. The study of monotonicity and negation complexity for Boolean functions has been prevalent in circuit complexity theory as well as in computational learning theory, but little attention has been given to it in the cryptographic context. Recently, Goldreich and Izsak [79] initiated a study of whether cryptographic primitives can be monotone, and showed that one-way functions can be monotone (assuming they exist), but a pseudorandom generator cannot.

In this chapter, we start by filling in the picture and proving that many other basic cryptographic primitives cannot be monotone. We then initiate a *quantitative* study of the power of negations, asking how many negations are required. We provide several lower bounds, many of them tight, for various cryptographic primitives and building blocks including one-way permutations, pseudorandom functions, small-bias generators, hard-core predicates, error-correcting codes, and randomness extractors.

Among our results, we show that, unlike one-way functions, one-way permutations cannot be monotone. Further, we prove that pseudorandom functions require $\log n - O(1)$ negations (which is optimal up to the additive term). Similarly, error-correcting codes with optimal distance parameters require $\log n - O(1)$ negations (again, optimal up to the additive term). Finally, we prove a general result for monotone functions, showing a lower bound on the depth of any circuit with t negations on the bottom that computes a monotone

function in terms of the monotone circuit depth complexity of the function.

1.4.3 Part III: Connections between Algorithms and Circuit lower bounds.

Chapters 3 and 4 provide strong lower bounds for restricted classes of Boolean circuits. Unfortunately, it is unclear how to adapt the methods described in these chapters in order to understand more general circuit classes.

In this part of the thesis, we study an alternative approach to unconditional lower bounds based on the design of faster algorithms.

Chapter 7: Constructing hard functions from learning algorithms. Fortnow and Klivans [70] proved the following relationship between efficient learning algorithms and circuit lower bounds: if a class $\mathcal{C} \subseteq \mathsf{P/poly}$ of Boolean circuits is exactly learnable with membership and equivalence queries in polynomial-time, then $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathcal{C}$. The class $\mathsf{EXP}^{\mathsf{NP}}$ was subsequently improved to EXP by Hitchcock and Harkins [92]. In this chapter, we improve on these results, and obtain the following consequences:

(*i*) If $\mathcal{C}$ is exactly learnable with membership and equivalence queries in polynomial-time, then $\mathsf{DTIME}(n^{\omega(1)}) \not\subseteq \mathcal{C}$. We obtain even stronger consequences if the class $\mathcal{C}$ is learnable in the mistake-bounded model, in which case we prove an average-case hardness result against $\mathcal{C}$.

(*ii*) If $\mathcal{C}$ is learnable in polynomial time in the PAC model then $\mathsf{PSPACE} \not\subseteq \mathcal{C}$, unless $\mathsf{PSPACE} \subseteq \mathsf{BPP}$. Removing this extra assumption from the statement of the theorem would provide an unconditional proof that $\mathsf{PSPACE} \not\subseteq \mathsf{BPP}$.

(*iii*) If $\mathcal{C}$ is efficiently learnable in the Correlational Statistical Query (CSQ) model, we show that there exists an explicit function f that is average-case hard for circuits in $\mathcal{C}$. To our knowledge, this result provides stronger average-case hardness guarantees than those obtained by SQ-dimension arguments (Blum et al. [30]). We also obtain a non-constructive extension of this result to the stronger Statistical Query (SQ) model.

Our proofs regarding exact and mistake-bounded learning are simple and self-contained, yield explicit hard functions, and show how to use mistake-bounded learners to "diagonal-

ize" over families of polynomial-size circuits. Our consequences for PAC learning lead to new proofs of Karp-Lipton-style collapse results, and the lower bounds from SQ learning make use of recent work relating combinatorial discrepancy to the existence of hard-on-average functions.

Chapter 8: Satisfiability algorithms, useful properties, and lower bounds. We prove that the existence of *non-trivial* satisfiability algorithms for certain classes of Boolean circuits of depth $d + 2$ leads to NEXP lower bounds for the corresponding class of circuits of depth d, where an algorithm is non-trivial if it runs in time $2^n/n^{\omega(1)}$ on polynomial size circuits over n input variables. Our proof simplifies and generalizes recent theorems obtained by Williams [198, 199], and extends the applicability of his techniques to certain classes of Boolean circuits not covered by his original results.

These results are connected to the notion of *useful properties* introduced in Williams' subsequent work [201]. Roughly speaking, a property of Boolean functions is *useful* if it is a *natural property* in the sense of Razborov and Rudich [154], but is not necessarily dense. We revisit Williams' connection, and show that the usual notion of *advice* from computational complexity plays a subtle role in the investigation of useful properties.

Finally, we discuss applications of these ideas to other frameworks connecting algorithms to circuit lower bounds, and introduce meta-connections between different frameworks of this form.

We state a few open problems in the final part of some chapters. We finish with some concluding remarks in Chapter 9.

Chapter 2

Preliminaries and Notation

We introduce below some basic results and definitions. We defer more specific definitions and technical background to each appropriate chapter. Recall that the Boolean circuit model was described in Section 1.3. We will revisit this model a few times, as we shift our attention to different classes of circuits.

Given a positive integer ℓ, we let $[\ell] \stackrel{\text{def}}{=} \{1, \ldots, \ell\}$. For a Boolean string w, we use $|w|$ to denote its length, and $|w|_1$ to denote the number of 1s in w.

The following standard concentration bound (cf. Alon and Spencer [13], Appendix A) will be useful.

Proposition 2.0.1. *Let $X_1, \ldots, X_m$ be independent $\{0,1\}$ random variables, where each X_i is 1 with probability $p \in [0,1]$. In addition, set $X \stackrel{\text{def}}{=} \sum_i X_i$, and $\mu \stackrel{\text{def}}{=} \mathbb{E}[X] = pm$. For any fixed $\zeta > 0$ there exists a constant $c_\zeta > 0$ independent of m and p such that*

$$\Pr[\,|X - \mu| > \zeta\mu] < 2e^{-c_\zeta \mu}.$$

A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is *balanced* (or unbiased) if $\Pr_x[f(x) = 1] = 1/2$, where x is uniformly distributed. The (total) *influence* (also know as average sensitivity) of f is defined as

$$\mathrm{Inf}(f) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \mathrm{Inf}_i(f), \quad \text{where} \quad \mathrm{Inf}_i(f) \stackrel{\text{def}}{=} \Pr_{x \in \{0,1\}^n}[f(x) \neq f(x^{\oplus i})]$$

and $x^{\oplus i}$ denotes x with its i-th coordinate flipped. The quantity $\mathrm{Inf}_i(f)$ is the influence of

the i-th variable of f. Moreover, the *noise stability* of f at rate $\eta \in [-1,1]$ is

$$\mathrm{Stab}_\eta(f) \stackrel{\text{def}}{=} 1 - 2\Pr[\, f(x) \neq f(y)\,],$$

where x is drawn uniformly at random from $\{0,1\}^n$, and y is obtained from x by independently for each bit having $\Pr[y_i = x_i] = (1+\eta)/2$ (i.e., x and y are η-*correlated*). It is more convenient in some situations to work with a closely related measure. We use $\mathsf{NS}_p(f)$ to denote the *noise sensitivity* of f under noise rate $p \in [0,1/2]$, which is defined as $\Pr[f(x \oplus y) \neq f(x)]$, where x is distributed uniformly over $\{0,1\}^n$, and y is the p-biased binomial distribution over $\{0,1\}^n$, i.e., each coordinate of y is set to 1 independently with probability p. We refer to O'Donnell [147] for further information about such complexity measures.

Recall that a function $f_n \colon \{0,1\}^n \to \{0,1\}$ is *monotone* if $f_n(x) \leq f_n(y)$ whenever $x \preceq y$, where $\preceq$ denotes the bitwise partial order on $\{0,1\}^n$. We use $\mathsf{depth}^+(f)$ to denote the minimum depth among all monotone fan-in two Boolean circuits computing f.

We review next the (monotone) Karchmer-Wigderson [114] correspondence between monotone Boolean circuits and communication protocols (we refer to Kushilevitz and Nisan [128] for a formal presentation). Let $f \colon \{0,1\}^m \to \{0,1\}$ be a monotone function. Consider the following communication game between two players. The first player ("Alice") is given an input $w_{\mathsf{yes}} \in f^{-1}(1)$, while the second player ("Bob") is given $w_{\mathsf{no}} \in f^{-1}(0)$. The goal of the players is to agree on a coordinate $i \in [m]$ for which $w_{\mathsf{yes}}(i) = 1$ and $w_{\mathsf{no}}(i) = 0$. Since f is monotone, the set of coordinates with this property is non-empty. Let $\mathsf{KW}^+(f)$ denote this communication game, and $\mathsf{CC}^{\mathsf{det}}(\mathsf{KW}^+(f))$ be the (worst-case) communication cost of the best (two-party) deterministic protocol that solves $\mathsf{KW}^+(f)$. The following equivalence holds.

Proposition 2.0.2. *Let $f \colon \{0,1\}^n \to \{0,1\}$ be a monotone function. There exists a monotone fan-in two Boolean circuit C_f of depth d that computes f if, and only if, there exists a protocol Π_f for the (monotone) KW-game of f with communication cost d. In other words,*

$$\mathsf{depth}^+(f) \;=\; \mathsf{CC}^{\mathsf{det}}(\mathsf{KW}^+(f)).$$

Given a language $L \subseteq \{0,1\}^*$, we let $L_n \stackrel{\text{def}}{=} L \cap \{0,1\}^n$. We view L_n as a Boolean function in the natural way.

Part I

Circuit Lower Bounds

Chapter 3

On the monotone complexity of the satisfiability problem

3.1 Background, results, and organization

In this chapter we investigate the monotone circuit depth complexity of constraint satisfaction problems (CSP). Before describing our contributions, we mention recent developments that directly relate to our work.

Monotone Circuit Complexity. Recall that a Boolean circuit C is a *monotone circuit* if it contains only (fan-in two) AND and OR gates. Monotone circuits compute precisely the class of monotone functions. We refer the reader to Jukna [109] for an introduction to Boolean circuits, and to the monograph written by Korshunov [125] for more background on monotone functions. The *depth* of a circuit C is the length of its longest path from the output gate to an input variable. Monotone circuits have been intensively investigated in circuit complexity. For the importance of this line of research, we refer to Raz and McKenzie [Section 1.1, 152].

Several *unconditional* lower bounds are known in monotone circuit complexity. Until last year, the strongest depth lower bound for an explicit monotone function was the result obtained by Raz and Wigderson [153] showing that the matching problem over m-vertex graphs requires monotone circuits of depth $\Omega(m)$. Observe that this statement provides a

sequence of Boolean functions over $O(n)$ inputs and with complexity $\Omega(\sqrt{n})$, since m-vertex graphs are represented with $\binom{m}{2}$ input bits. A new lower bound in monotone depth complexity was recently obtained by Göös and Pitassi [87]. They proved that there is a sequence of functions $f = \{f_n\}_{n\in\mathbb{N}}$ in NP that requires monotone circuits of depth $\Omega(n/\log n)$. This result essentially settles the problem of showing the existence of explicit monotone functions with very large monotone depth complexity.[1]

Constraint Satisfaction Problems and Parameterized Complexity. A *constraint satisfaction problem* $\phi = (\mathsf{Vars}(\phi), \mathsf{Const}(\phi))$ consists of a set of variables $\mathsf{Vars}(\phi)$, and a set of Boolean constraints $\mathsf{Const}(\phi)$. Each constraint $C \in \mathsf{Const}(\phi)$ is a function $C\colon [r]^{V_C} \to \{0,1\}$, where $V_C \subset \mathsf{Vars}(\phi)$ is the set of variables associated to C, and $r \in \mathbb{N}$. The set $[r]$ is the *domain* of the CSP (when $r = 2$, we usually identify $[2]$ with $\{0,1\}$). We say that a CSP ϕ is *satisfiable* if there exists an assignment $\alpha\colon \mathsf{Vars}(\phi) \to [r]$ such that $C(\alpha|_{V_C}) = 1$ for every $C \in \mathsf{Const}(\phi)$. We will focus here on k-uniform CSPs, i.e., those on which every set V_C contains exactly k elements. In other words, every constraint depends on k variables. Our results hold in more generality, but assuming uniformity simplifies the presentation considerably.

It is well-known that checking the satisfiability of general CSPs is NP-hard. Due to their generality and relevance, both in theory and in practice, there is a vast literature containing theoretical and experimental results on this computational problem. More related to our work are the results connecting the hardness of solving CSPs to the "geometry", or hypergraph, associated to its constraints. Put another way, one considers the incidence relation between constraints (edges) and variables (vertices), without taking into account the functions defining the constraints. This leads to more specific CSP instances defined over a class of hypergraphs, and to the investigation of the hardness of the satisfiability problem on these inputs. This research direction is related to the subfield of complexity theory that tries to understand the precise parameters of the input that make a computational problem hard ("Parameterized Complexity", cf. Flum and Grohe [67] and Downey and Fellows [58]).

[1] It is not hard to see that any monotone function over n input variables can be computed by monotone circuits of depth $O(n)$. A simple (non-constructive) counting argument establishes the existence of monotone functions that require depth $\Omega(n)$ even with respect to general (non-monotone) Boolean circuits.

A sequence of works culminating in the beautiful results of Grohe, Schwentick, and Segoufin [89], Grohe [90], and Marx [133] provides strong evidence that the complexity of solving these instances in the worst-case is essentially characterized by the *tree-width* of these hypergraphs.[2] Roughly speaking, the tree-width complexity of a (hyper)graph measures how similar to a tree is the graph, and satisfaction problems whose underlying geometry is a tree are known to admit polynomial time algorithms. This concept was introduced by a few different authors using distinct but equivalent formulations, and is by now a standard notion in graph theory and algorithm design (cf. Bodlaender [35]).

We remark that the hardness results mentioned above provide *conditional* lower bounds based on strong assumptions from parameterized complexity theory, and assume the domain size of the CSPs to be *unbounded*. The reader is referred to these references for a precise formulation of the results.

In this chapter we introduce an explicit connection between monotone circuit complexity and the investigation of constraint satisfaction problems in parameterized complexity. Among our contributions, we prove an *unconditional* lower bound in a widely investigated model of computation showing that the hardness of the satisfiability problem is closely related to the tree-width of the corresponding graphs.

We start our discussion with a specific circuit lower bound that follows from our framework, and that may be of independent interest. We then introduce a more general class of problems, parameterized by a hypergraph $\mathcal{H}$ and the domain size r of the CSP instances. We present general techniques to prove upper bounds and lower bounds on the monotone parallel complexity of these problems. These results lead to a general theorem that classifies, for a large class of satisfiability problems, those that admit depth-efficient monotone circuits. Finally, we discuss consequences of these results in algorithm design.

Remark 1. *Our asymptotic statements are taken with respect to $n \to \infty$, while all other parameters remain fixed. If $\mathfrak{A}$ is a mathematical structure (such as a graph, a vector, a finite function, etc.), we will use $\lceil \mathfrak{A} \rfloor$ to denote its representation as an appropriate*

[2] Recall that we are discussing k-uniform CSPs. For the general case of non-uniform hypergraphs, i.e., CSP instances on which distinct constraints may depend on a different number of variables, the complexity of the satisfiability problem is investigated in Marx [134].

string in $\{0,1\}^*$. *We use* $\mathsf{depth}^+(f)$ *to denote the monotone circuit depth complexity of a function* $f\colon \{0,1\}^m \to \{0,1\}$ *with respect to the usual model of fan-in two monotone circuits. When discussing graphs and hypergraphs, a calligraphic letter such as* $\mathcal{H}$ *always denote a hypergraph, while normal letters such as* G *will be used for undirected graphs.*

Lower Bounds. We consider the monotone circuit depth complexity of the classic NP-complete problem k-SAT: given a k-CNF ψ over n input variables, is it satisfiable? More precisely, we study the complexity of a natural encoding of k-SAT as a monotone Boolean function, which will be denoted by $k\text{-}\mathsf{SAT}_n^+$. For convenience, we refer to a clause C of a CNF ψ by a pair (A, b) consisting of its set of variables A together with a Boolean vector $\vec{b} \in \{0,1\}^A$ that encodes which variables appear negated in C.

Definition 3.1.1. *Let* $k \geq 2$, *and* $\mathcal{H}_n^{(k)} = (V_n, E_n)$ *be the complete* k*-uniform hypergraph on* n *vertices. We consider a Boolean function* $k\text{-}\mathsf{SAT}_n^+\colon \{0,1\}^{\binom{n}{k}\cdot 2^k} \to \{0,1\}$ *defined as follows. An instance* $\lceil \psi \rfloor$ *of* $k\text{-}\mathsf{SAT}_n^+$ *represents a* k-CNF ψ *over the set of variables* V_n, *and we let* $k\text{-}\mathsf{SAT}_n^+(\lceil \psi \rfloor) = 1$ *if and only if* ψ *is satisfiable. The input variables of this function are represented by* $\{C_{e,\vec{b}} \mid e \in E_n \text{ and } \vec{b} \in \{0,1\}^e\}$, *where* $C_{e,\vec{b}} = 0$ *if and only if the clause* $(e, \vec{b})$ *occurs in* ψ.

For example, the input $1^{\binom{n}{k}\cdot 2^k}$ represents a trivial instance of the satisfiability problem in which all clauses are "turned off", while the input $0^{\binom{n}{k}\cdot 2^k}$ is the unsatisfiable instance that "activates" every clause. Observe that $k\text{-}\mathsf{SAT}_n^+$ is a *monotone* encoding of the k-SAT problem using $O(n^k)$ input variables. It is possible to show that, for any fixed k, this function admits fan-in two monotone circuits of depth $O(n)$. We establish the following near-matching lower bound.

Theorem 3.1.2. *There exists* $k \in \mathbb{N}$ *for which the following holds:*

$$\mathsf{depth}^+(k\text{-}\mathsf{SAT}_n^+) = \Omega(n/\log n).$$

Recall that a monotone *formula* is a monotone Boolean circuit with internal gates of fan-out one (see Jukna [109] for more details). The *size* of a formula is defined as the number of internal gates of the circuit. It is known that monotone formulas of size s can be converted into monotone formulas of depth $O(\log s)$ (Spira [178] and Wegener [197]). Therefore, we get the following consequence of Theorem 3.1.2.

Corollary 3.1.3. *If $\{F_n\}_{n\in\mathbb{N}}$ is a sequence of monotone formulas computing k-SAT_n^+, then*

$$\mathsf{size}(F_n) = 2^{\Omega(n/\log n)}.$$

Theorem 3.1.2 follows from a related stronger result, and we switch now to a more general treatment of constraint satisfaction problems and monotone circuit complexity. For convenience, we say that $\phi = (\mathsf{Vars}(\phi), \mathsf{Const}(\phi))$ is an (n, m, k, r)-CSP if it is a k-uniform CSP with domain size r consisting of n variables and m constraints. The canonical computational problem associated to (n, m, k, r)-CSPs is the following: Given an arbitrary CSP ϕ of this form, does it admit a satisfying assignment α?

We formalize here the discussion presented above on the geometry of CSPs. Given a constraint satisfaction problem ϕ, there is a natural hypergraph associated to it, defined as follows.

Definition 3.1.4. *Let ϕ be an (n, m, k, r)-*CSP*. The* geometry *of ϕ is given by*

$$\mathsf{Geom}(\phi) \stackrel{\text{def}}{=} \{V_C \subset \mathsf{Vars}(\phi) \mid C \in \mathsf{Const}(\phi)\}.$$

In addition, we associate to ϕ the k-uniform hypergraph $\mathcal{H}_\phi$ with vertex set $V(\mathcal{H}_\phi) \stackrel{\text{def}}{=} \mathsf{Vars}(\phi)$ and edge set $E(\mathcal{H}_\phi) \stackrel{\text{def}}{=} \mathsf{Geom}(\phi)$, where $|V(\mathcal{H}_\phi)| = n$ and $|E(\mathcal{H}_\phi)| = m$.

Observe that the geometry of a CSP ϕ does not depend on its constraints or domain size. Fix a k-uniform hypergraph $\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H}))$ on n vertices and m edges, and a domain size r. We consider a (further) restriction of the satisfiability problem for (n, m, k, r)-CSPs by focusing on instances with geometry $\mathcal{H}$. This formulation of the satisfiability problem has a natural encoding as a monotone Boolean function. More precisely, each constraint can be specified by r^k input bits, leading to the following definition.

Definition 3.1.5. *Let $r \in \mathbb{N}$ be a fixed parameter. Given a k-uniform hypergraph $\mathcal{H}$ on n vertices and m edges, we consider a restriction of the (n, m, k, r)-*CSP *satisfiability problem to $\mathcal{H}$, encoded as a Boolean function*

$$\mathsf{CSP}_{\mathcal{H},r} \colon \{0,1\}^{\mathcal{L}} \to \{0,1\},$$

where $\mathcal{L} = \{(e, \beta) \mid e \in E(\mathcal{H}) \text{ and } \beta \in [r]^e\}$. Given an input $\lceil\phi\rfloor \in \{0,1\}^{\mathcal{L}}$, we let $\mathsf{CSP}_{\mathcal{H},r}(\lceil\phi\rfloor) \stackrel{\text{def}}{=} 1$ if and only if there exists an assignment $\alpha \in [r]^{V(\mathcal{H})}$ such that, for every $e \in E(\mathcal{H})$, we have $\lceil\phi\rfloor(e, \alpha|_e) = 1$.

Notice that $\beta = \alpha|_e$ is the restriction of the assignment α to the variables in e, and that the input bits of $\mathsf{CSP}_{\mathcal{H},r}$ indexed by $(e, \cdot)$ encode the constraint over these variables. In particular, if $\lceil \phi \rfloor$ is an input string, $\lceil \phi \rfloor(e, \beta) = 1$ indicates that the partial assignment β satisfies this constraint. Therefore, $\mathsf{CSP}_{\mathcal{H},r}(\lceil \phi \rfloor) = 1$ if and only if the CSP instance $\phi = (\mathsf{Vars}(\phi), \mathsf{Const}(\phi))$ encoded by $\lceil \phi \rfloor$ is satisfiable.

One can assume an ordering of $\mathcal{L}$, and view $\mathsf{CSP}_{\mathcal{H},r}$ as a Boolean function over variables $x_1, \ldots, x_N$, where $N = mr^k$. It is easy to see that $\mathsf{CSP}_{\mathcal{H},r}$ is a monotone Boolean function, since if $\lceil \phi_1 \rfloor \preceq \lceil \phi_2 \rfloor$ over the N-dimensional Boolean hypercube, then any satisfying assignment for ϕ_1 is also a satisfying assignment for ϕ_2. This is because each constraint of ϕ_2 can be viewed as a relaxation of the corresponding constraint of ϕ_1.

Our next theorem can be seen as a refinement of the Göös-Pitassi lower bound, and provides a hard function with a more natural description in the spirit of the lower bound proved by Raz and Wigderson. (Recall that a hypergraph $\mathcal{H}$ is *ℓ-regular* if every vertex $v \in V(\mathcal{H})$ is contained in exactly ℓ edges.)

Theorem 3.1.6. *There exist $k \in \mathbb{N}$ and an explicit sequence $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ of 2-regular k-uniform hypergraphs on $nk/2$ vertices and n edges such that*

$$\mathsf{depth}^{+}(\mathsf{CSP}_{\mathcal{H}_n,3}) = \Omega(n/\log n).$$

Constraint satisfaction problems for which the number of constraints is linear in the number of variables play a fundamental role in many results and hardness assumptions (see e.g. Coja-Oghlan [51], Impagliazzo, Paturi, and Zane [103], and Feige [62]). Theorem 3.1.6 provides unconditional evidence of the computational hardness of this class of instances.

Most monotone depth lower bounds are obtained via a connection to communication complexity discovered by Karchmer and Wigderson [114] and M. Yannakakis (unpublished). We follow the same strategy here. The proof of Theorem 3.1.6 is based on an extension of a technique introduced by Raz and McKenzie [152] and further developed by Göös and Pitassi [Theorem 5, 87]. It allows us to reduce our circuit lower bound to a certain lower bound in communication complexity involving a related (composed) search problem.

One of our key insights is that it is possible to preserve the geometry of the CSPs involved in this reduction. This is formalized in the more abstract Theorem 3.1.8, presented

next. We will get back to the overview of the proof of Theorem 3.1.6 after we describe this new tool.

Definition 3.1.7. *Given an* (n, m, k, r)-CSP $\phi = (\mathsf{Vars}(\phi), \mathsf{Const}(\phi))$*, we consider its* canonical search problem $\mathsf{S}(\phi)$*, which is the relation* $\mathsf{S}(\phi) \subseteq [r]^{\mathsf{Vars}(\phi)} \times \mathsf{Const}(\phi)$ *defined as follows:*

$$\mathsf{S}(\phi) \stackrel{\text{def}}{=} \{(\alpha, C) \mid \alpha \in [r]^{\mathsf{Vars}(\phi)},\, C \in \mathsf{Const}(\phi), \text{ and } C(\alpha|_{V_C}) = 0\}.$$

If ϕ is unsatisfiable, then for every α there exists a "solution" C such that $(\alpha, C) \in \mathsf{S}(\phi)$. Put another way, given an assignment α, one has to provide a clause that is *not* satisfied by α.

In general, a *search problem* is a relation $S \subseteq [\ell]^n \times Q$, where Q is a set of possible solutions. Given an input $\alpha \in [\ell]^n$, we let $S(\alpha) \stackrel{\text{def}}{=} \{q \in Q \,|\, (\alpha, q) \in S\}$ denote the set of solutions of α in S. All search problems discussed here will be *total*, i.e., $S(\alpha) \neq \emptyset$ for all $\alpha \in [\ell]^n$.

Given a search problem $S \subseteq [\ell]^n \times Q$, and a function $g\colon \mathcal{X} \times \mathcal{Y} \to [\ell]$ over finite sets $\mathcal{X}$ and $\mathcal{Y}$, we consider a communication game between two players, defined as follows. The first player receives $\vec{x} \in \mathcal{X}^n$, while the second player gets $\vec{y} \in \mathcal{Y}^n$. We let $\alpha = g^{(n)}(\vec{x}, \vec{y}) \in [\ell]^n$ be the vector with $\alpha_i \stackrel{\text{def}}{=} g(x_i, y_i)$. The goal of the players is to agree on a solution $q \in S(\alpha)$. We use $\mathsf{CC}^{\mathsf{det}}(S \circ g^{(n)})$ to denote the two-party deterministic communication complexity of this (composed) search problem in the standard communication complexity model (cf. Kushilevitz and Nisan [128]).

As mentioned above, we obtain the following refinement of a technique employed by Raz and McKenzie [152] and Göös and Pitassi [87].

Theorem 3.1.8. *Let* $\mathcal{H} = (V, E)$ *be a* k*-uniform hypergraph, where* $k \geq 2$ *and* $|V| = n$*, and let* $g\colon \mathcal{X} \times \mathcal{Y} \to [\ell]$*, where* $r \stackrel{\text{def}}{=} \min\{|\mathcal{X}|, |\mathcal{Y}|\} \geq 2$ *and* $\ell \geq 2$*. Then,*

$$\mathsf{depth}^{+}(\mathsf{CSP}_{\mathcal{H},r}) \;\geq\; \max_{\lceil \phi \rfloor \in \mathsf{CSP}^{-1}_{\mathcal{H},\ell}(0)} \mathsf{CC}^{\mathsf{det}}\big(\mathsf{S}(\phi) \circ g^{(n)}\big). \tag{3.1}$$

Roughly speaking, this result says that in order to lower bound the depth complexity of solving constraint satisfaction problems with geometry $\mathcal{H}$, we can investigate communication games associated to fixed *unsatisfiable* CSPs over the same geometry. Observe that during

this reduction the domain size changes from r to ℓ (we will apply this result with a function g for which $r > \ell$).

We are now in position to discuss the remaining ideas employed in the proof of Theorem 3.1.6, which follows the argument used by Göös and Pitassi [87], while controlling the hypergraphs constructed during the different stages of the proof.

Huynh and Nordström [99] discovered a method to lower bound $\mathsf{CC}^{\mathsf{det}}\big(\mathsf{S}(\phi) \circ g_{\star}^{(n)}\big)$ based on a complexity measure of the search problem $\mathsf{S}(\phi)$ called *critical block sensitivity*.[3] Their approach works with a specific function $g_{\star} \colon [3] \times \{0,1\}^3 \to \{0,1\}$ whose definition will not be important in our proof. Since we are concerned with a class of search problems defined with respect to a fixed hypergraph $\mathcal{H}$, i.e., we can consider any $\lceil \phi \rfloor \in \mathsf{CSP}^{-1}_{\mathcal{H},2}(0)$ in Theorem 3.1.8, it will be more convenient to describe their method in connection with our results. For this, we define a new complexity measure for hypergraphs based on their notion of sensitivity for Boolean relations.

Let $S \subseteq \{0,1\}^{[n]} \times Q$ be a total search problem, and $f \subseteq S$ be a total function. In other words, $f \colon \{0,1\}^{[n]} \to Q$, and for every $\alpha \in \{0,1\}^{[n]}$, it is the case that $f(\alpha) \in S(\alpha)$. We use $\mathsf{Tot}(S)$ to denote the set of total functions $f \subseteq S$. The *block sensitivity* of f at α will be denoted by $\mathsf{bs}(f, \alpha)$. More precisely, this is the maximum number k of disjoint sets of input variables $B_1, \ldots, B_k \subseteq [n]$ such that $f(\alpha) \neq f(\alpha^{B_i})$ for every $i \in [k]$, where α^{B_i} denotes the string $\alpha^{B_i} \in \{0,1\}^{[n]}$ obtained by flipping the value of every bit indexed by B_i. An input $\alpha \in \{0,1\}^{[n]}$ is said to be *critical* if $|S(\alpha)| = 1$. We use $\mathsf{Crit}(S)$ to denote the set of critical inputs $\alpha \in S$. The *critical block sensitivity* of a total search problem $S \subseteq \{0,1\}^{[n]} \times Q$ is given by

$$\mathsf{bs}_{\mathsf{crit}}(S) \stackrel{\text{def}}{=} \min_{f \in \mathsf{Tot}(S)} \max_{\alpha \in \mathsf{Crit}(S)} \mathsf{bs}(f, \alpha).$$

Observe that the critical block sensitivity of a relation is a more or less natural extension of the notion of block sensitivity from Boolean functions, modulo the restriction to critical inputs (check Buhrman and de Wolf [42] for more on Boolean function complexity measures).

Huynh and Nordström proved among their results that, for any total search problem

[3] An alternative proof of their result with slightly different parameters can be found in the work of Göös and Pitassi [87]. Using the corresponding theorem from [87] would only change a few constants in our statements.

S as above, we have

$$\mathsf{CC}^{\mathsf{det}}\big(S \circ g_\star^{(n)}\big) = \Omega(\mathsf{bs}_{\mathsf{crit}}(S)). \tag{3.2}$$

This lower bound and Theorem 3.1.8 motivate the following definition.

Definition 3.1.9. *Let $\mathcal{H}$ be a hypergraph. The* (CSP *critical block*) sensitivity *of $\mathcal{H}$ is given by*

$$\mathsf{sens}(\mathcal{H}) \stackrel{\text{def}}{=} \max_{\lceil\phi\rfloor \in \mathsf{CSP}^{-1}_{\mathcal{H},2}(0)} \mathsf{bs}_{\mathsf{crit}}(\mathsf{S}(\phi)).$$

In other words, the sensitivity of a hypergraph $\mathcal{H}$ depends on structural properties of *unsatisfiable* CSPs with domain size 2 and geometry $\mathcal{H}$.

We can now combine the communication lower bound of Huynh and Nordström [99] based on critical block sensitivity and Theorem 3.1.8 into the following result, which follows immediately from Equations (3.1) and (3.2), and the fact that $g_\star \colon [3] \times \{0,1\}^3 \to \{0,1\}$.

Corollary 3.1.10. *Let $\mathcal{H}$ be a k-uniform hypergraph, where $k \geq 2$. Then,*

$$\mathsf{depth}^+(\mathsf{CSP}_{\mathcal{H},3}) \;\geq\; \Omega(\mathsf{sens}(\mathcal{H})). \tag{3.3}$$

Observe that the right-hand side of inequality (3.3) has no reference to circuit complexity or communication complexity. In other words, we have the following "transfer principle" for the monotone circuit depth complexity of constraint satisfaction problems:

> "*The monotone depth complexity of* $\mathsf{CSP}_{\mathcal{H},3}$ *can be lower bounded by structural properties of Boolean-valued unsatisfiable* CSPs *with geometry* $\mathcal{H}$."

The proof of Theorem 3.1.6 follows from an application of this principle to a certain sequence of hypergraphs constructed from expander graphs in connection with ideas from proof complexity. This is an adaptation of the proof of [Theorem 3, 87], and we describe the details in Section 3.3.

Upper Bounds. It is a well-known phenomenon in algorithm design that several optimization problems that are hard for arbitrary graphs become easy when the inputs are restricted to special classes of graphs, such as acyclic graphs. It is natural therefore to search for a distance measure between a general graph and the class of acyclic graphs, and to try to adapt

algorithms for acyclic graphs to work with more general instances. A powerful realization of this idea appears in the study of the *tree-width* of graphs and hypergraphs, together with the design of efficient algorithms for hypergraphs of moderate tree-width complexity (see e.g. Gottlob, Greco, and Scarcello [Chapter 1, 37]).

Our main contribution in this direction is formulating a connection between this technique and *monotone* circuit depth complexity. Given a hypergraph $\mathcal{H} = (V(\mathcal{H}), E(\mathcal{H}))$, we consider a certain tree-width-based measure that is related to the monotone circuit depth complexity of the satisfiability problem for instances with geometry $\mathcal{H}$. For the reader familiar with the usual notion of tree-width of a hypergraph, we consider the tree decomposition of $\mathcal{H}$ that minimizes the *product* between the *width* of the decomposition, and the *depth* of the tree. We call this measure the *depth-width* complexity of $\mathcal{H}$, and denote this quantity by $\mathsf{depth\text{-}width}(\mathcal{H})$. A precise formulation is deferred to Section 3.4 (Definition 3.4.8), since this is a somewhat technical concept. It follows from a result of Bodlaender [34] that depth-width and tree-width are related by a logarithmic factor (see Section 3.5 for more details).

The next result shows that depth-width is closely connected to the monotone parallel complexity of constraint satisfaction problems.

Theorem 3.1.11. *Let $r, k \in \mathbb{N}$, and let $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be a sequence of connected k-uniform hypergraphs with $|V(\mathcal{H}_n)| = n$. Then,*

$$\mathsf{depth}^{+}(\mathsf{CSP}_{\mathcal{H}_n,r}) = O_{r,k}(\mathsf{depth\text{-}width}(\mathcal{H}_n)).$$

Theorem 3.1.11 is proved using dynamic programming over the best tree decomposition of $\mathcal{H}$. In order to implement this strategy, we need to make sure that all steps can be done using monotone circuits. Furthermore, these steps must be implemented efficiently in parallel. The depth-width complexity of the hypergraph guarantees that this is possible, allowing us to compute $\mathsf{CSP}_{\mathcal{H},r}$ via shallow monotone circuits. Since we have not defined depth-width, we defer further details about the proof of Theorem 3.1.11 to the body of the text.

Theorem 3.1.11 provides a general technique to prove upper bounds on the monotone circuit depth complexity of satisfiability problems. For instance, a straightforward appli-

cation of this result, together with an upper bound on the depth-width complexity of the Cycle (Appendix 3.6), leads to the following corollary.

Corollary 3.1.12. *Let C_n be the cycle on n vertices. Then, for any $r \in \mathbb{N}$,*

$$\mathsf{depth}^{+}(\mathsf{CSP}_{C_n,r}) = \Theta(\log n).$$

An unconditional classification theorem for CSPs. We say that a computational problem admits *depth-efficient* algorithms if it can be solved by a sequence of circuits of poly-logarithmic depth in the size of the input. One of the main technical contributions of this chapter is the proof of a classification theorem that describes, for a large class of satisfiability problems, those that admit depth-efficient monotone circuits.

Given an undirected k-regular graph G, we obtain a k-uniform hypergraph $\mathcal{H}$ from G as follows. The edges of G become vertices of $\mathcal{H}$, and each hyperedge of $\mathcal{H}$ corresponds to the set of edges connecting a vertex $v \in V(G)$ to all its neighbors in G. We say that $\mathcal{H}$ is the *Tseitin* hypergraph of G, and write $\mathcal{H} = \mathsf{Tseitin}(G)$. A precise definition and more intuition for this construction can be found in Section 3.3. Using the machinery described above, we establish the following general result.

Theorem 3.1.13. *There exists a fixed constant $c > 0$ for which the following holds. If $\{G_n\}_{n\in\mathbb{N}}$ is a sequence of undirected graphs, where each G_n is a k-regular connected graph on n vertices, then*

$$\Omega(\mathsf{tree\text{-}width}(G_n)^c) \;\leq\; \mathsf{depth}^{+}\big(\mathsf{CSP}_{\mathsf{Tseitin}(G_n),3}\big) \;\leq\; O_k(\mathsf{tree\text{-}width}(G_n)\cdot \log n).$$

Theorem 3.1.13 shows in a certain technical sense that monotone depth complexity and tree-width are polynomially related complexity measures.

Corollary 3.1.14. *Let $\{G_n\}_{n\in\mathbb{N}}$ be a sequence of undirected graphs, where each G_n is a k-regular connected graph on n vertices, and $k \in \mathbb{N}$. Then,*

$$\mathsf{depth}^{+}\big(\mathsf{CSP}_{\mathsf{Tseitin}(G_n),3}\big) \;\leq\; (\log n)^{O(1)} \quad\Longleftrightarrow\quad \mathsf{tree\text{-}width}(G_n) \;\leq\; (\log n)^{O(1)}.$$

In other words, $\mathsf{CSP}_{\mathsf{Tseitin}(G),3}$ admits depth-efficient monotone algorithms if and only if G is reasonably close to a tree. This is similar to a result of Grohe [90] from parameterized

complexity that provides a characterization of the classes of constraint satisfaction problems that can be solved by polynomial time algorithms. Observe, however, that Theorem 3.1.13 gives an *unconditional* lower bound. Intuitively, Grohe's result relies on a hardness assumption from parameterized complexity that is implied by the Exponential Time Hypothesis (Impagliazzo and Paturi [100]). Since Corollary 3.1.3 can be seen as establishing a weaker form of this conjecture for monotone formulas, it is reasonable that we should be able to prove a result similar to Theorem 3.1.13 using the techniques described above.

Nevertheless, this is only a high-level abstraction. Formally, the upper bound follows from a lemma that bounds the tree-width of $\mathsf{Tseitin}(G)$ by a function of the tree-width of G, combined with the tight connection between tree-width and depth-width, and an application of Theorem 3.1.11. On the other hand, the lower bound relies mainly on Corollary 3.1.10, the techniques used to establish Theorem 3.1.6, and an improvement of the Grid-Minor Theorem obtained by Chekuri and Chuzhoy [48]. Interestingly, there are similarities between our approach, which gives an unconditional circuit lower bound, and Grohe's conditional proof.

Discussion and Further Remarks. Our main conceptual contribution is the introduction of a general framework to investigate the monotone complexity of satisfiability problems. In particular, our results establish the existence of *natural* computational problems with $\Omega(n/\log n)$ monotone depth complexity. It would be interesting to close the gap between this lower bound and the trivial $O(n)$ upper bound.

From a technical perspective, this chapter provides methods that can be used to prove lower bounds (Theorem 3.1.8) and upper bounds (Theorem 3.1.11) on the monotone circuit depth complexity of constraints satisfaction problems based on their hypergraphs. These techniques are strong enough to imply a classification result that describes the approximate complexity of a general class of CSP instances (Theorem 3.1.13). To our knowledge, this is the first hardness result of this form for CSPs of *bounded-size domain* (even among conditional lower bounds).

Our work shows that general constraint satisfaction problems cannot be solved efficiently in parallel using techniques that yield "monotone algorithms" (Theorem 3.1.6).

This includes a large body of work in parameterized complexity that relies on treewidth-based decompositions. The same lower bound holds for satisfaction problems over sparse hypergraphs, i.e., instances with a linear number of constraints.

From an algorithmic perspective, this implies that if one hopes to construct a parallel algorithm for satisfiability with super-logarithmic savings over the trivial $O(n)$ depth upper bound, some *non-monotone* computation must be employed. While monotone circuits can be substantially weaker than non-monotone circuits when computing monotone functions (see e.g. Tardos [183], Ajtai and Gurevich [7], Hofmeister [97]), it is unclear to us whether a similar phenomenon happens with respect to the parallel complexity of satisfiability problems.

We believe that our ideas will lead to other unconditional lower bounds for monotone circuits. In particular, it would be interesting to establish a tighter and more general characterization of the hardness of constraint satisfaction problems based on their hypergraphs, in analogy to the results of Marx [133].

Organization of the Chapter. The proof of Theorem 3.1.8 appears in Section 3.2. We derive Theorems 3.1.2 and 3.1.6 from this result and other techniques in Section 3.3. The definition of the depth-width complexity of a hypergraph and the proof of Theorem 3.1.11 are presented in Section 3.4. In Section 3.5, we provide the proof of Theorem 3.1.13. Finally, Section 3.6 describes a low complexity depth-width tree decomposition.

3.2 A transfer principle for constraint satisfaction problems

In this section we prove Theorem 3.1.8. This completes the proof of Corollary 3.1.10, which will be used in Section 3.3 to derive our lower bounds. First, we describe the *minterms* of $\mathsf{CSP}_{\mathcal{H},r}$.[4] While this is not strictly necessary for the proof of Theorem 3.1.8, it sheds light into the structure of the Boolean function $\mathsf{CSP}_{\mathcal{H},r}$, and familiarizes the reader with our notation.

[4]Recall that $x \in \{0,1\}^m$ is a *minterm* of a monotone Boolean function $g\colon \{0,1\}^m \to \{0,1\}$ provided that $g(x) = 1$ and, whenever $y \preceq x$ with $y \neq x$, we have $g(y) = 0$.

Lemma 3.2.1. *Let $\mathcal{H} = (V, E)$ be a k-uniform hypergraph, and $N = |E(\mathcal{H})| \cdot r^k$. Then $\lceil \phi \rfloor \in \{0,1\}^N$ is a minterm of $\mathsf{CSP}_{\mathcal{H},r}$ if and only if*

(*i*) *$\phi = (\mathsf{Vars}(\phi), \mathsf{Const}(\phi))$ is satisfiable; and*

(*ii*) *For every constraint $C \in \mathsf{Const}(\phi)$, where $C \colon [r]^{\mathsf{Vars}(C)} \to \{0,1\}$, we have $|C^{-1}(1)| = 1$.*

Proof. Clearly, if conditions (*i*) and (*ii*) are satisfied, we have $\mathsf{CSP}_{\mathcal{H},r}(\lceil \phi \rfloor) = 1$. Further, if $\lceil \psi \rfloor \prec \lceil \phi \rfloor$ (strictly) under these conditions, then some constraint of ψ admits no (partial) satisfying assignment. In particular, $\mathsf{CSP}_{\mathcal{H},r}(\lceil \psi \rfloor) = 0$, and it follows that $\lceil \phi \rfloor$ is a minterm.

On the other hand, for any minterm $\lceil \phi \rfloor$, there exists an assignment $\alpha \colon V(\mathcal{H}) \to [r]$ such that $\phi(\alpha) = 1$, given that $\lceil \phi \rfloor$ is a 1-input of $\mathsf{CSP}_{\mathcal{H},r}$. In addition, it must be the case that for every constraint $C \in \mathsf{Const}(\phi)$, $|C^{-1}(1)| = 1$, since otherwise $\lceil \phi \rfloor$ would not be a minterm under our encoding of ϕ by $\lceil \phi \rfloor$. □

We say that a hypergraph $\mathcal{H} = (V, E)$ is *covered by its edges* if $\bigcup_{e \in E(\mathcal{H})} e = V(\mathcal{H})$.

Lemma 3.2.2. *Let $\mathcal{H} = (V, E)$ be a k-uniform hypergraph that is covered by its edges, and $N = |E(\mathcal{H})| \cdot r^k$. Consider the following sets of minterms and assignments, respectively:*

$$\begin{aligned} \mathcal{M} &\stackrel{\text{def}}{=} \{ \lceil \phi \rfloor \in \{0,1\}^N \mid \lceil \phi \rfloor \text{ is a minterm of } \mathsf{CSP}_{\mathcal{H},r} \}, \\ \mathcal{A} &\stackrel{\text{def}}{=} \{ \alpha \mid \alpha \colon V(\mathcal{H}) \to [r] \}. \end{aligned}$$

There exists a (canonical) bijection $\zeta \colon \mathcal{M} \to \mathcal{A}$.

Proof. It follows from Lemma 3.2.1 and the fact that $\mathcal{H}$ is covered by its edges that every minterm $\lceil \phi \rfloor$ of $\mathsf{CSP}_{\mathcal{H},r}$ has a *unique* satisfying assignement α_ϕ. On the other hand, using once again that $\mathcal{H}$ is covered by its edges, every assignment $\alpha \colon V(\mathcal{H}) \to [r]$ gives rise to a *unique* constraint satisfaction problem ϕ_α with $|C^{-1}(1)| = 1$ for every $C \in \mathsf{Const}(\phi_\alpha)$ that is satisfied by α. The map ζ is then obtained by setting $\zeta(\lceil \phi \rfloor) = \alpha_\phi$. It is clear that ζ is a bijection between the sets $\mathcal{M}$ and $\mathcal{A}$. □

Lemma 3.2.2 shows that there is a natural correspondence between the minterms of $\mathsf{CSP}_{\mathcal{H},r}$ and r-valued assignments for the variables in $V(\mathcal{H})$ (assuming that $\mathcal{H}$ is covered by

its edges). Since the set $\mathcal{M}_g$ of minterms of a monotone Boolean function g gives rise to a monotone DNF with $|\mathcal{M}_g|$ terms that computes g, it is easy to see that

$$\mathsf{depth}^{+}(\mathsf{CSP}_{\mathcal{H},r}) \;=\; O\big(\log r^{|V(\mathcal{H})|}\big) \;=\; O\big(|V(\mathcal{H})| \cdot \log r\big),$$

as one would expect.

The minterms of $\mathsf{CSP}_{\mathcal{H},r}$ play a crucial role in the proof of Theorem 3.1.8, stated again for convenience of the reader.

Theorem. *Let $\mathcal{H} = (V, E)$ be a k-uniform hypergraph, where $k \geq 2$ and $|V| = n$, and let $g\colon \mathcal{X} \times \mathcal{Y} \to [\ell]$, where $r \stackrel{\text{def}}{=} \min\{|\mathcal{X}|, |\mathcal{Y}|\} \geq 2$ and $\ell \geq 2$. Then,*

$$\mathsf{depth}^{+}(\mathsf{CSP}_{\mathcal{H},r}) \;\geq\; \max_{\lceil \phi \rfloor \in \mathsf{CSP}^{-1}_{\mathcal{H},\ell}(0)} \mathsf{CC}^{\mathsf{det}}\big(\mathsf{S}(\phi) \circ g^{(n)}\big).$$

Proof. Let D be a monotone Boolean circuit of depth d computing $\mathsf{CSP}_{\mathcal{H},r}$. Using the Karchmer-Wigderson connection (Proposition 2.0.2), it is possible to solve the corresponding communication game $\mathsf{KW}^{+}(\mathsf{CSP}_{\mathcal{H},r})$ by a protocol Π with communication cost d. We use Π to solve the (composed) search problem $\mathsf{S}(\phi) \circ g^{(n)}$, where ϕ is an arbitrary ℓ-valued *unsatisfiable* constraint satisfaction problem with geometry $\mathcal{H}$ and variable set V, i.e., $\lceil \phi \rfloor \in \mathsf{CSP}^{-1}_{\mathcal{H},\ell}(0)$. This is sufficient to establish Theorem 3.1.8. We remark that the unsatisfiability of ϕ is necessary to guarantee that in the reduction the players produce 1-inputs and 0-inputs, respectively, for the monotone Karchmer-Wigderson game of $\mathsf{CSP}_{\mathcal{H},r}$. More details follow.

Fix an unsatisfiable ϕ, as described above. Recall that in the communication game $\mathsf{S}(\phi) \circ g^{(n)}$ Alice is given $x \in \mathcal{X}^{\mathsf{Vars}(\phi)}$, Bob is given $y \in \mathcal{Y}^{\mathsf{Vars}(\phi)}$, $\alpha \stackrel{\text{def}}{=} g^{(n)}(x, y) \in [\ell]^{\mathsf{Vars}(\phi)}$, and the players must agree on a constraint $C \in \mathsf{Const}(\phi)$ such that $C(\alpha|_{\mathsf{Vars}(C)}) = 0$. Assume without loss of generality that $r = |\mathcal{X}|$, and let $\pi\colon \mathcal{X} \to [r]$ be a fixed bijection between these sets. Roughly speaking, we identify $\mathcal{X}$ with $[r]$ during the remaining of the proof.[5] For convenience, we assume that all CSP instances discussed below are over the same set of input variables $V = \mathsf{Vars}(\phi)$. The domain of each CSP is not necessarily the same.

[5]Whenever we apply π or π^{-1} to a vector, it means that we are applying the corresponding permutation to each coordinate of the vector.

Given x, Alice produces a positive instance $\lceil \psi_x \rfloor$ of $\mathsf{CSP}_{\mathcal{H},r}$ as follows. Consider a constraint $C \in \mathsf{Const}(\psi_x)$, where $C \colon [r]^{\mathsf{Vars}(C)} \to \{0,1\}$, and let $\beta \in [r]^{\mathsf{Vars}(C)}$ be an input to C, i.e., $\beta \colon \mathsf{Vars}(C) \to [r]$. Alice sets $C(\beta) = 1$ in $\lceil \psi_x \rfloor$ if and only if β agrees with her input $x \in \mathcal{X}^V$ under π, that is, if for every variable $w \in \mathsf{Vars}(C)$, $\pi(x(w)) = \beta(w)$. This completes the definition of ψ_x. Observe that $\lceil \psi_x \rfloor$ is a minterm of $\mathsf{CSP}_{\mathcal{H},r}$ (according to Lemma 3.2.1) that is satisfied by $\lambda \stackrel{\text{def}}{=} \pi(x)$. Further, by construction, ψ_x and ϕ have the same geometry $\mathcal{H}$ (actually, ψ_x depends only on $\mathcal{H}$ and not on ϕ).

Given y, Bob produces a negative instance $\lceil \psi_y \rfloor$ of $\mathsf{CSP}_{\mathcal{H},r}$, defined as follows. Consider a constraint $C \in \mathsf{Const}(\psi_y)$, and let β be as above. Bob sets $C(\beta) = 1$ in $\lceil \psi_y \rfloor$ if and only if $\gamma \stackrel{\text{def}}{=} g^{(d)}\big(\pi^{-1}(\beta), y|_{\mathsf{Vars}(C)}\big) \in [\ell]^{\mathsf{Vars}(C)}$ is a partial assignment to the variables in V that satisfies the corresponding constraint C' of ϕ defined over $\mathsf{Vars}(C)$ (observe that by definition ϕ and ψ_y have the same geometry $\mathcal{H}$). This completes the definition of ψ_y. We claim that $\mathsf{CSP}_{\mathcal{H},r}(\lceil \psi_y \rfloor) = 0$, i.e., ψ_y is unsatisfiable. Assume otherwise that some $\lambda \in [r]^V$ satisfies ψ_y, and let $\gamma \stackrel{\text{def}}{=} g^{(n)}(\pi^{-1}(\lambda), y) \in [\ell]^V$. Then, using the definition of ψ_y, every constraint C' of ϕ is satisfied by γ. This contradicts our initial assumption that ϕ is unsatisfiable.

The previous reduction uses no communication. The parties now run protocol Π over inputs $\lceil \psi_x \rfloor$ and $\lceil \psi_y \rfloor$, respectively, and obtain a coordinate $i \in [N]$ such that $\lceil \psi_x \rfloor_i = 1$ and $\lceil \psi_y \rfloor_i = 0$, where $N = |\mathcal{H}(E)| \cdot r^d$ is the number of input bits of the Boolean function $\mathsf{CSP}_{\mathcal{H},r}$. By definition, i is a coordinate associated to a fixed constraint C, corresponding to an edge of $\mathcal{H}$. We claim that the corresponding constraint C of ϕ is violated by $\alpha = g^{(n)}(x,y)$. This, however, follows from the fact that $i \in [N]$ is mapped to a fixed $\beta \in [r]^{\mathsf{Vars}(C)}$, and from the corresponding definitions of $C(\beta)$ in ψ_x and ψ_y, respectively, as functions of x, y, and ϕ. To sum up, Alice and Bob are able to solve the search problem $\mathsf{S}(\phi) \circ g^{(n)}$ with communication cost d, which completes the proof. □

Observe that Theorem 3.1.8 allows us to prove lower bounds against r-valued CSPs, where r depends on g. Observe that $r = 3$ in the statement of Corollary 3.1.10. We will see in Section 3.3 that such results can be translated into depth lower bounds for Boolean-valued CSPs as well, such as the one given by Theorem 3.1.2.

3.3 Lower bounds for k-SAT and sparse CSPs

In this section we present the proofs of Theorems 3.1.2 and 3.1.6. As explained before, we rely on Corollary 3.1.10, which reduces our circuit lower bounds to the study of structural properties of hypergraphs and their associated (unsatisfiable) CSPs. First, we give the argument for Theorem 3.1.6. Then we explain how to derive Theorem 3.1.2 from this circuit lower bound via a sequence of monotone reductions.

Given a graph or hypergraph $\mathcal{H} = (V, E)$ and a vertex $v \in V$, we let

$$N_{\mathcal{H}}(v) \stackrel{\text{def}}{=} \{u \in V \mid \exists e \in E \text{ such that } \{v, u\} \subseteq e\} \setminus \{v\}$$

be the set of neighbors of v in $\mathcal{H}$ (excluding v). Further, it will be convenient for our proof to let

$$\mathsf{deg}_{\mathcal{H}}(v) \stackrel{\text{def}}{=} |\{e \in E \mid v \in e\}|$$

be the number of edges of $\mathcal{H}$ containing v, i.e., the degree of v in $\mathcal{H}$.

Recall that we identify the domain set $[2] = \{1, 2\}$ of instances of $\mathsf{CSP}_{\mathcal{H},2}$ with the set $\{0, 1\}$. The next two definitions show how to define a Boolean CSP from an undirected graph and a coloring of its vertices.

Definition 3.3.1. *Let $G = (V_G, E_G)$ be an undirected graph. The* Tseitin hypergraph *of G, denoted by $\mathsf{Tseitin}(G)$, is the hypergraph $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$ defined as follows:*

$$V_{\mathcal{H}} \stackrel{\text{def}}{=} \{v_e \mid e = \{u, w\} \text{ is an edge of } G\} \quad \text{and} \quad E_{\mathcal{H}} \stackrel{\text{def}}{=} \{e_u \mid u \in V_G \text{ and } e_u = \downarrow(N_G(u))\},$$

where $\downarrow(N_G(u)) \stackrel{\text{def}}{=} \{v_{\{u,w\}} \mid w \in N_G(u)\}$.

Put another way, the edges of G become vertices of the hypergraph, and the hyperedges are defined based on the neighborhood sets of G. For the reader familiar with graph theory terminology, the Gaifman graph of the hypergraph $\mathsf{Tseitin}(G)$ is simply the line graph of G.

Definition 3.3.2. *Let $G = (V_G, E_G)$ be an undirected graph, and $\chi \colon V_G \to \{0, 1\}$ be a coloring of V_G. Further, let $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$ be the Tseitin hypergraph of G. We use $\phi_G^{\chi} = (\mathsf{Vars}(\phi_G^{\chi}), \mathsf{Const}(\phi_G^{\chi}))$ to denote the following instance of $\mathsf{CSP}_{\mathcal{H},2}$: for each $e_v \in E_{\mathcal{H}}$, where $v \in V_G$, the corresponding constraint*

$$C_{e_v}(\beta) \stackrel{\text{def}}{=} 1 \iff \sum_{w \in e_v} \beta(w) = \chi(v) \ (\mathsf{mod}\ 2),$$

where $\beta \in \{0,1\}^{e_v}$.

In the language of the original graph G, this constraint satisfaction problem asks for the existence of a Boolean assignment to the edges of G that respects the parity constraint on each vertex of G. A simple parity argument gives the following result (check [Section 4.1, 86] for more details).

Fact 3.3.3. *For a connected graph* G, ϕ_G^χ *is* unsatisfiable *if and only if* $|\chi^{-1}(1)|$ *is odd. In other words, if this is the case then* $\lceil \phi_G^\chi \rfloor \in \mathsf{CSP}^{-1}_{\mathcal{H},2}(0)$, *where* $\mathcal{H} = \mathsf{Tseitin}(G)$.

The next lemma contains the observation that the Tseiting hypergraph is uniform and 2-regular if the original undirected graph is regular.

Lemma 3.3.4. *Let* $G = (V_G, E_G)$ *be a* k*-regular undirected graph, and* $\mathcal{H} = \mathsf{Tseitin}(G)$, *where* $\mathcal{H} = (V_\mathcal{H}, E_\mathcal{H})$. *Then* $\mathcal{H}$ *is a* k*-uniform hypergraph, and for every* $v_e \in V_\mathcal{H}$, $\mathsf{deg}_\mathcal{H}(v_e) = 2$.

Proof. It is easy to see that $\mathcal{H}$ is k-uniform. The remaining claim is also clear from Definition 3.3.1, since every vertex $v_e \in V_\mathcal{H}$ comes from some edge $e \in E_G$ with $e = \{u, w\}$, and the only edges in $E_\mathcal{H}$ containing v_e are $e_u = \downarrow(N_G(u))$ and $e_w = \downarrow(N_G(w))$. □

We will apply Corollary 3.1.10 to a (Tseitin) hypergraph $\mathcal{H}$ constructed from an (explicit) expander graph G. The reason is that there is a connectivity parameter of undirected graphs called routing number that can be used to lower bound the critical block sensitivity of the related search problem.

More precisely, for an undirected graph $G = (V, E)$, we say that G is *k-routable* if there exists a set $T \subseteq V$ of size $2k$ such that for any set of k disjoint pairs of nodes of T, there are k *edge-disjoint* paths in G that connect such pairs. In addition, we let $\mathsf{routing}(G)$ (the *routing number* of G) be the largest k such that G is k-routable.

Proposition 3.3.5 (Göös and Pitassi [87])**.** *Let* $G = (V, E)$ *be a connected graph,* $\chi \colon V \to \{0,1\}$ *be a function for which* $|\chi^{-1}(1)|$ *is odd, and* ϕ_G^χ *be the corresponding (unsatisfiable) instance of* $\mathsf{CSP}_{\mathcal{H},2}$, *where* $\mathcal{H} = \mathsf{Tseitin}(G)$. *Then,*

$$\mathsf{bs}_{\mathsf{crit}}(\mathsf{S}(\phi_G^\chi)) \geq \mathsf{routing}(G).$$

The next proposition guarantees the existence of sparse regular graphs with large routing number.

Proposition 3.3.6 (Frieze [74])**.** *There exists $k \in \mathbb{N}$ and an explicit sequence $\{G_n\}_{n\in\mathbb{N}}$ of connected k-regular graphs on n vertices for which*

$$\mathsf{routing}(G_n) = \Omega(n/\log n).$$

These results allow us to prove the following central lemma.

Lemma 3.3.7. *There exists $k \in \mathbb{N}$ and an explicit sequence $\{H_n\}_{n\in\mathbb{N}}$ of k-uniform 2-regular hypergraphs on $nk/2$ vertices and n edges such that*

$$\mathsf{sens}(\mathcal{H}_n) = \Omega(n/\log n).$$

Proof. Let $\{G_n\}_{n\in\mathbb{N}}$ be the sequence of graphs given by Proposition 3.3.6, where $G_n = (V_{G_n}, E_{G_n})$. Let $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be the corresponding sequence of Tseitin hypergraphs, where $\mathcal{H}_n = \mathsf{Tseitin}(G_n)$. Observe that each $\mathcal{H}_n$ has $nk/2$ vertices and n edges. Further, Lemma 3.3.4 guarantees that each $\mathcal{H}_n$ is k-uniform and 2-regular. Now fix some sequence of functions $\chi_n \colon V_{G_n} \to \{0,1\}$, where $|\chi_n^{-1}(1)|$ is odd. Fact 3.3.3 implies that each CSP $\phi_{G_n}^{\chi_n}$ is unsatisfiable. Put another way, $\lceil \phi_{G_n}^{\chi_n} \rfloor \in \mathsf{CSP}^{-1}_{\mathcal{H}_n,2}(0)$. Further, Proposition 3.3.5 implies that

$$\mathsf{bs}_{\mathsf{crit}}(\mathsf{S}(\phi_{G_n}^{\chi_n})) \geq \mathsf{routing}(G_n) = \Omega(n/\log n). \tag{3.4}$$

Finally, using the definition of sensitivity of a hypergraph (Definition 3.1.9) and inequality (3.4), we get that $\mathsf{sens}(\mathcal{H}_n) = \Omega(n/\log n)$, which completes the proof. □

We are now in position to prove Theorem 3.1.6, stated again below.

Theorem. *There exist $k \in \mathbb{N}$ and an explicit sequence $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ of 2-regular k-uniform hypergraphs on $nk/2$ vertices and n edges such that*

$$\mathsf{depth}^{+}(\mathsf{CSP}_{\mathcal{H}_n,3}) = \Omega(n/\log n).$$

Proof. The result is immediate from Corollary 3.1.10 and Lemma 3.3.7. □

We proceed with the proof of Theorem 3.1.2. As mentioned before, this result is obtained via a sequence of monotone reductions. Our first lemma shows that k-SAT_n^+ is closely related to our more general instances of the satisfiability problem.

Lemma 3.3.8. *For $k \geq 2$, let $\mathcal{H}_n^{(k)}$ be the complete k-uniform hypergraph on n vertices. Then,*

$$\mathsf{depth}^+(k\text{-}\mathsf{SAT}_n^+) = \mathsf{depth}^+\big(\mathsf{CSP}_{\mathcal{H}_n^{(k)},2}\big).$$

Proof. We observe that these satisfiability problems correspond to the same Boolean function under a permutation of the input variables. The argument is easier to understand via an example of the correspondence between constraints from $\mathsf{CSP}_{\mathcal{H}_n^{(2)},2}$ and clauses in 2-SAT_n^+. Consider a constraint $C^{\phi}_{\{x_1,x_2\}}$ of an instance $\lceil \phi \rfloor$ of $\mathsf{CSP}_{\mathcal{H}_n^{(2)},2}$, as shown in Figure 3.1.

x_1	x_2	$C^{\phi}_{\{x_1,x_2\}}(x_1,x_2)$
0	0	1
0	1	0
1	0	1
1	1	0

Figure 3.1: The truth-table of a Boolean-valued constraint $C^{\phi}_{\{x_1,x_2\}}$ represented by the 2-CNF $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$.

The canonical 2-CNF representation of the Boolean function over two input variables that computes this constraint is given by $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$. Crucially, the monotone encoding of this 2-CNF according to our definition of 2-SAT_n^+ is given by $C_{\{1,2\},(0,0)} = 1$, $C_{\{1,2\},(0,1)} = 0$, $C_{\{1,2\},(1,0)} = 1$, and $C_{\{1,2\},(1,1)} = 0$. In general, the reduction between these two satisfiability problems relies on this bijection involving contraints and clauses. In other words, every block of input variables coming from a constraint in $\mathsf{CSP}_{\mathcal{H}_n^{(k)},2}$ is mapped to the corresponding 2^k clauses in k-SAT_n^+, and vice-versa. The only thing to observe is that this map can be done monotonically.

More formally, for each constraint C_e with $e \in \mathcal{H}_n^{(k)}$, where $C_e \colon [2]^e \to \{0,1\}$, and each $\beta \in [2]^e$, we associate the input variable in the block of variables C_e indexed by β with

the input variable $C_{(e,\beta')}$ of $k\text{-}\mathsf{SAT}_n^+$, where $\beta'(v) \stackrel{\text{def}}{=} \beta(v) - 1$ for every $v \in e$, i.e., β' is the $\{0,1\}$-version of $\beta \in [2]^e$. This is clearly a bijection between the input variables of each problem. In addition, using the canonical k-CNF computing each constraint as shown in the example, and the definition of this map, it is not hard to see that the original CSP instance is satisfiable if and only if the resulting k-CNF is satisfiable. Finally, the reduction is monotone, which completes the proof. □

The next lemma formalizes the intuitive fact that if $\mathcal{H}_1 \subseteq \mathcal{H}_2$, then the satisfiability problem for instances with geometry $\mathcal{H}_1$ cannot be harder than the same problem for $\mathcal{H}_2$.

Lemma 3.3.9. *Let $k \geq 2$ be a fixed integer, and consider k-uniform hypergraphs $\mathcal{H}_1 = (V_1, E_1)$ and $\mathcal{H}_2 = (V_2, E_2)$ with $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. Then, for every integer $r \geq 2$,*

$$\mathsf{depth}^+(\mathsf{CSP}_{\mathcal{H}_2,r}) \geq \mathsf{depth}^+(\mathsf{CSP}_{\mathcal{H}_1,r})$$

Proof. It is easy to see that an input $\lceil \phi_1 \rfloor$ of $\mathsf{CSP}_{\mathcal{H}_1,r}$ can be projected to a corresponding input $\lceil \phi_2 \rfloor$ of $\mathsf{CSP}_{\mathcal{H}_2,r}$ such that ϕ_1 is satisfiable if and only if ϕ_2 is satisfiable. The reduction may need to set to 1 some input variables of $\mathsf{CSP}_{\mathcal{H}_2,r}$ that do not correspond to any constraint coming from the geometry $\mathcal{H}_1$. However, given a monotone circuit D for $\mathsf{CSP}_{\mathcal{H}_2,r}$, these constants 1 can be used to simplify D accordingly, providing a monotone circuit for $\mathsf{CSP}_{\mathcal{H}_1,r}$ of at most the same depth. □

A similar argument yields the following result, which states that a decrease on the domain size cannot increase monotone circuit depth complexity.

Lemma 3.3.10. *For any uniform hypergraph $\mathcal{H}$ and positive integers $r > r' \geq 2$, we have*

$$\mathsf{depth}^+(\mathsf{CSP}_{\mathcal{H},r}) \geq \mathsf{depth}^+(\mathsf{CSP}_{\mathcal{H},r'}).$$

Finally, we have a lemma that allows us to reduce the domain size of the CSP instances without sacrificing the monotone complexity of the problem, at the cost of producing a slightly more complex geometry.

Lemma 3.3.11. *Let $\mathcal{H} = (V, E)$ be a k-uniform hypergraph with $n = |V|$ and $m = |E|$. Assume that $r = 2^d$, where $d \in \mathbb{N}$, and let $n' = dn$, $m' = m$, and $k' = dk$. Consider the*

k'-uniform hypergraph $\mathcal{H}' = (V', E')$ on n' vertices and m' edges defined as follows.

$$\begin{aligned} V' &\stackrel{\text{def}}{=} \{v'_i \mid v \in V \text{ and } i \in [d]\}, \text{ and} \\ E' &\stackrel{\text{def}}{=} \{e' \mid \exists e \in E \text{ such that } e' = \bigcup_{v \in e} \{v'_1, \ldots, v'_k\}\} \end{aligned}$$

Then $\mathsf{depth}^+(\mathsf{CSP}_{\mathcal{H}',2}) \geq \mathsf{depth}^+(\mathsf{CSP}_{\mathcal{H},r})$.

Proof. We observe the existence of a monotone projection from $\mathsf{CSP}_{\mathcal{H},r}$ to $\mathsf{CSP}_{\mathcal{H}',2}$. Given an instance $\lceil \phi \rfloor$ of the former, this projection produces an input $\lceil \phi' \rfloor$ of the latter such that

$$\phi \text{ is satisfiable} \iff \phi' \text{ is satisfiable.} \tag{3.5}$$

Consider a block of input variables C_e of $\mathsf{CSP}_{\mathcal{H},r}$, for some $e \in E$. More precisely, we view $C_e^\phi \colon [r]^e \to \{0,1\}$ as a fixed function (for a given ϕ), and C_e as the set of Boolean variables encoding this function. Let $\beta \in [r]^e$ be the index of an input variable in C_e. In order to describe ϕ', we map this variable to a corresponding variable of $C_{e'}$, where $e' \in E'$ is the edge of $\mathcal{H}'$ obtained from e. It is sufficient to describe the index $\beta' \in [2]^{e'}$ corresponding to β, and to prove that condition (3.5) is true.

We sketch the construction next. Each variable $v \in \mathsf{Vars}(\phi) = V$ is mapped to the set of variables $\{v'_1, \ldots, v'_d\} \subseteq \mathsf{Vars}(\phi') = V'$, and each individual assignment $a \in [r] = [2^d]$ is mapped in a one-to-one manner to a fixed assignment $(b_1, \ldots, b_d) \in [2]^d$. This induces a natural map between the truth-table of C_e^ϕ and the truth-table of $C_{e'}^{\phi'}$. Further, there is a bijection between the satisfying assignments $\alpha \in [r]^V$ of ϕ and the satisfying assignments $\alpha' \in [2]^{V'}$ of ϕ'. In particular, condition (3.5) holds. Finally, observe that ϕ' is constructed using a monotone projection, as each β is mapped to a corresponding β', which completes the proof. □

Observe that Lemma 3.3.11 reduces the size of the domain of the instances at the cost of increasing the arity of the contraints. Notice that $N = m \cdot r^k = m' \cdot 2^{k'} = N'$, and therefore $\mathsf{CSP}_{\mathcal{H},r} \colon \{0,1\}^N \to \{0,1\}$ and $\mathsf{CSP}_{\mathcal{H}',2} \colon \{0,1\}^{N'} \to \{0,1\}$ are monotone functions on the same number of input bits.

We have all ingredients to give the proof of Theorem 3.1.2.

Theorem. *There exists $k \in \mathbb{N}$ for which the following holds:*

$$\mathsf{depth}^{+}(k\text{-}\mathsf{SAT}^{+}_{n}) = \Omega(n/\log n).$$

Proof. The result follows from Theorem 3.1.6 and the reductions formalized in Lemmas 3.3.8, 3.3.9, 3.3.10, and 3.3.11. More precisely, let $\{\mathcal{H}^{\star}_{n}\}_{n\in\mathbb{N}}$ be the sequence of $k^{\star}$-uniform hypergraphs on $nk^{\star}/2$ vertices granted by Theorem 3.1.6. Then,

$$\begin{aligned}
\Omega(n/\log n) &\overset{\text{Theorem 3.1.6}}{=} \mathsf{depth}^{+}\big(\mathsf{CSP}_{\mathcal{H}^{\star}_{n},3}\big)\\
&\overset{\text{Lemma 3.3.10}}{\leq} \mathsf{depth}^{+}\big(\mathsf{CSP}_{\mathcal{H}^{\star}_{n},4}\big)\\
&\overset{\text{Lemma 3.3.11}}{\leq} \mathsf{depth}^{+}\big(\mathsf{CSP}_{\mathcal{H}_{n},2}\big)\\
&\overset{\text{Lemma 3.3.9}}{\leq} \mathsf{depth}^{+}\big(\mathsf{CSP}_{\mathcal{H}^{(k)}_{n},2}\big)\\
&\overset{\text{Lemma 3.3.8}}{=} \mathsf{depth}^{+}\big(k\text{-}\mathsf{SAT}^{+}_{nk^{\star}}\big),
\end{aligned}$$

where $\mathcal{H}_{n}$ is obtained from $\mathcal{H}^{\star}_{n}$ using Lemma 3.3.11, $k = 2k^{\star}$, and $\mathcal{H}^{(k)}_{n}$ is the complete k-uniform hypergraph on $nk^{\star}$ vertices. The theorem follows since we have $k^{\star} = O(1)$. □

3.4 Upper bounds via depth-width complexity

This section is dedicated to the proof of Theorem 3.1.11. We view a constraint $C\colon [r]^{\mathsf{Vars}(C)} \to \{0,1\}$ as a family of assignments $C \subseteq [r]^{\mathsf{Vars}(C)}$. We will use $\lceil C \rfloor$ to denote the Boolean vector in $\{0,1\}^{r^{|\mathsf{Vars}(C)|}}$ that encodes this family. For convenience, given $\beta\colon \mathsf{Vars}(C) \to [r]$, we address the bit position in $\lceil C \rfloor$ corresponding to β by $\lceil C \rfloor_{\beta}$. In other words, $C(\beta) = 1$ if and only if $\lceil C \rfloor_{\beta} = 1$.

Definition 3.4.1. *Given two families of assignments $A_1 \subseteq [r]^{V}$ and $A_2 \subseteq [r]^{U}$, we let the* (*relational*) join *of A_1 and A_2, denoted by $A_1 \bowtie A_2$, be the family of assignments $B \subseteq [r]^{V\cup U}$ defined by*

$$B = \{\beta\colon V \cup U \to [r] \mid \beta|_{V} \in A_1 \text{ and } \beta|_{U} \in A_2\}.$$

Observe that the join operation is associative and commutative.

Fact 3.4.2. *Given constraints $C_1\colon [r]^{V} \to \{0,1\}$ and $C_2\colon [r]^{U} \to \{0,1\}$, the set of assignments mapping $V \cup U$ to $[r]$ satisfying both C_1 and C_2 is precisely $C_1 \bowtie C_2$.*

It follows from Fact 3.4.2 that a CSP $\phi = (\mathsf{Vars}(\phi), \mathsf{Const}(\phi))$ is satisfiable if and only if

$$\bowtie_{C \in \mathsf{Const}(\phi)} C \neq \emptyset. \tag{3.6}$$

Lemma 3.4.3. *Given constraints $C_1 \colon [r]^{\mathsf{Vars}(C_1)} \to \{0,1\}$ and $C_2 \colon [r]^{\mathsf{Vars}(C_2)} \to \{0,1\}$, there exists a (multi-output) monotone fan-in two Boolean circuit of depth 1 that computes $\lceil C_1 \bowtie C_2 \rfloor$ from $\lceil C_1 \rfloor$ and $\lceil C_2 \rfloor$.*

Proof. For convenience, let $V \stackrel{\text{def}}{=} \mathsf{Vars}(C_1)$ and $U \stackrel{\text{def}}{=} \mathsf{Vars}(C_2)$, and $\beta \colon V \cup U \to [r]$ be an index to an output wire of the Boolean circuit. Then, according to Definition 3.4.1,

$$\lceil C_1 \bowtie C_2 \rfloor_\beta = \lceil C_1 \rfloor_{\beta|_V} \wedge \lceil C_2 \rfloor_{\beta|_U}.$$

In other words, every bit of $\lceil C_1 \bowtie C_2 \rfloor$ is computed from $\lceil C_1 \rfloor$ and $\lceil C_2 \rfloor$ by a monotone fan-in two depth 1 circuit. □

Observe that Lemma 3.4.3 together with Equation 3.6 can be used to construct a monotone circuit that checks whether a CSP instance from a fixed geometry is satisfiable. This construction, however, is quite inefficient, as the number of output bits of the final join operation can be exponential. In particular, a disjunction over these many bits will have linear depth in the number of input variables of the CSP. We need to introduce other ideas in order to design a more efficient monotone circuit for this task.

Definition 3.4.4. *Given a family of assignments $A \subseteq [r]^V$ and a set $U \subseteq V$, we let the* projection *of A to U, denoted by $\pi_U(A)$, be the family of assignments $B \subseteq [r]^U$ defined by*

$$B = \{\beta \colon U \to [r] \mid \exists \alpha \in A \text{ such that } \alpha|_U = \beta\}.$$

Fact 3.4.5. *Given constraints $C_1 \colon [r]^V \to \{0,1\}$ and $C_2 \colon [r]^U \to \{0,1\}$, the set of assignments mapping $W \stackrel{\text{def}}{=} V \cap U$ to $[r]$ that can be* extended *to an assignment satisfying both C_1 and C_2 is precisely $\pi_W(C_1 \bowtie C_2)$.*

Lemma 3.4.6. *Given a constraint $C \colon [r]^V \to \{0,1\}$ and a set $U \subseteq V$, there exists a (multi-output) monotone fan-in two Boolean circuit of depth $d = O(|V \backslash U| \cdot \log r)$ that computes $\lceil \pi_U(C) \rfloor$ from $\lceil C \rfloor$.*

Proof. Let $\beta\colon U \to [r]$ be an index to an output wire of the Boolean circuit. Then, according to Definition 3.4.4,

$$\lceil \pi_U(C) \rfloor_\beta = \bigvee_{\substack{\alpha \in [r]^V \\ \alpha|_U = \beta}} \lceil C \rfloor_\alpha .$$

In other words, every bit of $\lceil \pi_U(C) \rfloor$ is computed from $\lceil C \rfloor$ by a monotone fan-in two circuit of depth $O(\log r^{|V \setminus U|}) = O(|V \setminus U| \cdot \log r)$. □

By combining these observations and the corresponding constructions, we obtain the following result.

Proposition 3.4.7. *Let U and V be fixed sets, and $r \in \mathbb{N}$. Given constraints $C_1\colon [r]^V \to \{0,1\}$ and $C_2\colon [r]^U \to \{0,1\}$, the set of assignments from $W \stackrel{\text{def}}{=} V \cap U$ to $[r]$ that can be extended to an assignment that satisfies both C_1 and C_2 can be computed by a (multi-output) fan-in two monotone Boolean circuit D of depth*

$$d = O\big(|(V \cup U) \setminus W| \cdot \log r\big).$$

In other words, for any input pair $(\lceil C_1 \rfloor, \lceil C_2 \rfloor) \in \{0,1\}^{r^{|V|} + r^{|U|}}$, we have

$$D(\lceil C_1 \rfloor, \lceil C_2 \rfloor) = \lceil \pi_W(C_1 \bowtie C_2) \rfloor \in \{0,1\}^{r^{|W|}}.$$

In order to solve $\mathsf{CSP}_{\mathcal{H},r}$ more efficiently, we will perform joins and projections in a carefully chosen order, and involving more general constraints. We will need a few additional definitions (cf. Flum and Grohe [Section 11.1, 67]).

We will be concerned with *trees* $\mathcal{T} = (T, F)$ that are rooted, directed, and binary. In other words, every node $t \in T$ has at most two immediate *successors*, and the set of such successors will be denoted by $S(t)$. A *leaf* $t \in T$ satisfies $S(t) = \emptyset$. For convenience, when manipulating trees and graphs simultaneously, we use T for the set of *nodes* of the three, and F for its set of *arcs*, i.e., we reserve *vertices* and *edges* for undirected graphs and hypergraphs. Given $t \in T$, we use $\mathcal{T}_t$ to denote the subtree of $\mathcal{T}$ rooted at t, i.e., T_t is the set of nodes reachable from t (including t), and F_t the corresponding set of arcs. A subset of nodes $S \subseteq T$ is *connected* if these vertices are connected in the underlying *undirected* tree.

Given a tree $\mathcal{T} = (T, F)$, the *depth* (also known as *rank*) of a node $t \in T$ is defined inductively as follows. If t is a leaf in $\mathcal{T}$, then $\mathsf{depth}(t) \stackrel{\text{def}}{=} 0$. Otherwise, $\mathsf{depth}(t) \stackrel{\text{def}}{=} 1 + \max_{t' \in S(t)} \mathsf{depth}(t')$. The depth of $\mathcal{T}$, denoted by $\mathsf{depth}(\mathcal{T})$, is the depth of its root node, or equivalently, the length of the longest (directed) path in $\mathcal{T}$ (as measured by its number of arcs).

For a hypergraph $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}})$, we let its *Gaifman graph* be the undirected graph $G = (V, E)$ with vertex set $V = V_{\mathcal{H}}$ and edge set $E = \{\{u, v\} \mid \exists e \in E_{\mathcal{H}} \text{ for which } u, v \in e\}$. In other words, the (hyper)edges of $\mathcal{H}$ become cliques in G. We say that $\mathcal{H}$ is *connected* if $\mathsf{Gaifman}(\mathcal{H})$ is connected.

Observe that $\mathsf{CSP}_{\mathcal{H},r}$ is easier for disconnected hypergraphs $\mathcal{H}$. In this case, one can simply solve the satisfiability problem corresponding to each connected component of $\mathsf{Gaifman}(\mathcal{H})$, since no constraint involves variables from different components. For this reason, we focus on connected hypergraphs from now on.

Definition 3.4.8. *A* tree decomposition *of a connected hypergraph $\mathcal{H} = (V, E)$ is a pair $(\mathcal{T}, \{B_t\}_{t \in T})$, where $\mathcal{T} = (T, A)$ is a tree, $\{B_t\}_{t \in T}$ is a family of subsets of V, and the following holds:*

(i) For every $v \in V$, the set $B^{-1}(v) \stackrel{\text{def}}{=} \{t \in T \mid v \in B_t\}$ is nonempty and connected in $\mathcal{T}$.

(ii) For every (hyper)edge $e \in E$, there exists a node $t \in T$ such that $e \subseteq B_t$.

The width *of a tree decomposition $\mathcal{D}_{\mathcal{H}} = (\mathcal{T}, \{B_t\}_{t \in T})$ is defined as*

$$\mathsf{width}(\mathcal{D}_{\mathcal{H}}) \stackrel{\text{def}}{=} \max_{t \in T} |B_t| - 1,$$

and its depth is defined as

$$\mathsf{depth}(\mathcal{D}_{\mathcal{H}}) \stackrel{\text{def}}{=} \mathsf{depth}(\mathcal{T}).$$

We say that a hypergraph $\mathcal{H}$ admits a (d, w)-depth-width decomposition if $\mathcal{H}$ admits a tree decomposition $\mathcal{D}_{\mathcal{H}}$ such that $\mathsf{depth}(\mathcal{D}_{\mathcal{H}}) \leq d$ and $\mathsf{width}(\mathcal{D}_{\mathcal{H}}) \leq w$. The family of all tree decompositions of a hypergraph $\mathcal{H}$ will be denoted by

$$\mathscr{D}(\mathcal{H}) \stackrel{\text{def}}{=} \{\mathcal{D} \mid \mathcal{D} = (\mathcal{T}, \{B_t\}_{t \in T}) \text{ is a tree decomposition of } \mathcal{H}\}.$$

The tree-width *of a hypergraph is given by*

$$\mathsf{tree\text{-}width}(\mathcal{H}) \stackrel{\text{def}}{=} \min_{\mathcal{D} \in \mathscr{D}(\mathcal{H})} \mathsf{width}(\mathcal{D}).$$

Finally, we define the depth-width *of a hypergraph $\mathcal{H}$ by*

$$\mathsf{depth\text{-}width}(\mathcal{H}) \stackrel{\text{def}}{=} \min_{\mathcal{D} \in \mathscr{D}(\mathcal{H})} \mathsf{depth}(\mathcal{D}) \cdot \mathsf{width}(\mathcal{D}).$$

We have included an example of a tree decomposition in Section 3.6. Notice that we defined depth-width directly for hypergraphs, as they constitute our main objects of study in connection with CSPs. Equivalently, one can define depth-width for graphs (2-uniform hypergraphs), and consider the Gaifman graph of a hypergraph (see e.g. Flum and Grohe [Proposition 11.27, 67]).

The proof of Theorem 3.1.11 requires a few additional definitions. Given a tree decomposition $\mathcal{D}_{\mathcal{H}} = (\mathcal{T}, \{B_t\}_{t \in T})$ of a hypergraph $\mathcal{H}$, we say that each set B_t is a *bag* of the decomposition. For a subset $U \subseteq T$, we let

$$B(U) \stackrel{\text{def}}{=} \bigcup_{t \in U} B_t.$$

Given a tree $\mathcal{T} = (T, F)$, we say that a node $u \in T$ is a *descendant* of a node $v \in T$, written $u \preceq v$, if there is a directed path from v to u in $\mathcal{T}$. Equivalently, u is contained in the subtree $\mathcal{T}_v$ rooted at v. The correctness of our monotone circuit relies on the following simple but fundamental property of tree decompositions.

Lemma 3.4.9. *Let $\mathcal{D}_{\mathcal{H}} = (\mathcal{T}, \{B_t\}_{t \in T})$ be a tree decomposition of a connected hypergraph $\mathcal{H} = (V, E)$, where $\mathcal{T} = (T, F)$, and consider a node $v \in T$ with set of successors $S(v) = \{u_1, u_2\}$. Let $\mathcal{T}_{u_1} = (T_{u_1}, F_{u_1})$ and $\mathcal{T}_{u_2} = (T_{u_2}, F_{u_2})$ be the corresponding subtrees rooted at u_1 and u_2, respectively. Then,*

$$B(T_{u_1}) \cap B(T_{u_2}) \subseteq B(v).$$

Proof. Let $w \in B_{T_{u_1}} \cap B_{T_{u_2}}$. Then there exist $v_1 \in T_{u_1}$ and $v_2 \in T_{u_2}$ such that $w \in B_{v_1}$ and $w \in B_{v_2}$. Since $\mathcal{T}$ is a tree, there exists a unique path from v_1 to v_2 in the undirected graph of $\mathcal{T}$. In particular, this path must pass through v. From the first property of Definition 3.4.8, we must have $w \in B_v$. Therefore, $B_{T_{u_1}} \cap B_{T_{u_2}} \subseteq B_v$, which completes the proof. □

We are ready to give the proof of Theorem 3.1.11, stated again below.

Theorem. *Let $r, k \in \mathbb{N}$, and let $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be a sequence of connected k-uniform hypergraphs with $|V(\mathcal{H}_n)| = n$. Then,*

$$\mathsf{depth}^{+}(\mathsf{CSP}_{\mathcal{H}_n,r}) = O_{r,k}(\mathsf{depth\text{-}width}(\mathcal{H}_n)).$$

Proof. Let $\mathcal{D} = (\mathcal{T}, \{B_t\}_{t\in T})$ be a tree decomposition of $\mathcal{H} = (V, E)$ with $\mathsf{depth\text{-}width}(\mathcal{D}) = \mathsf{depth\text{-}width}(\mathcal{H})$, where $\mathcal{T} = (T, F)$. In addition, let $w \stackrel{\text{def}}{=} \mathsf{width}(\mathcal{D})+1$ and $d \stackrel{\text{def}}{=} \mathsf{depth}(\mathcal{D})+1$. We will rely on the structure of $\mathcal{T}$ in order to construct a monotone circuit M computing $\mathsf{CSP}_{\mathcal{H},r}$. The core of the construction consists of d stages. During the i-th stage, we handle all nodes of $\mathcal{T}$ with depth strictly less than i, for $1 \leq i \leq d$. We use then the output of stage d in order to compute $\mathsf{CSP}_{\mathcal{H},r}(\lceil\phi\rfloor)$, where $\lceil\phi\rfloor \in \{0,1\}^{|E(\mathcal{H}_n)|\cdot r^k}$ is the input string.[6]

Recall than an input $\lceil\phi\rceil$ encodes a $\mathsf{CSP}_{\mathcal{H},r}$ instance $\phi = (\mathsf{Vars}(\phi), \mathsf{Const}(\phi))$, where $\mathsf{Vars}(\phi) = V$ and $\mathsf{Const}(\phi) = \{C_e^{\phi} \colon [r]^e \to \{0,1\} \mid e \in E\}$. For convenience, we will address the corresponding block of input variables by C_e, and view it as a family of assignments as given by Definitions 3.4.1 and 3.4.4. After the i-th stage of the construction of M, we will have the following invariant:

> ($\star$) For every $v \in T$ with $\mathsf{depth}(v) < i$, there exists a set $G_v = \{g_\beta^v\}_{\beta\in[r]^{B_v}}$ of at most r^w gates in the circuit M such that g_β^v evaluates to 1 on an input $\lceil\phi\rfloor$ if and only if β can be extended to a partial assignment $\alpha \in [r]^{B(T_v)}$ satisfying every constraint C_e^{ϕ} with $e \subseteq B_u$ for some $u \in T_v$.

First, let's see how to complete the description of our monotone circuit M assuming that ($\star$) holds for every i, and prove its correctness. Let $t \in T$ be the root of $\mathcal{T}$. We claim that

$$\phi \text{ is satisfiable} \iff \bigvee_{\beta\in[r]^{B_t}} g_\beta^t(\lceil\phi\rfloor) = 1. \tag{3.7}$$

If ϕ is satisfiable, let $\alpha \in [r]^V$ be a satisfying assignment for ϕ. Then, since $\beta \stackrel{\text{def}}{=} \alpha|_{B_t} \in [r]^{B_t}$ can be extended to a satisfying assignment, it follows from ($\star$) that $g_\beta^t(\lceil\phi\rfloor) = 1$. Consequently, the RHS of (3.7) evaluates to 1. On the other hand, if there exists $\beta \in [r]^{B_t}$

[6] The reader should not confuse the parameter d related to the depth of the tree decomposition with the final depth of the circuit M, which will be larger than d.

for which $g_\beta^t(\lceil\phi\rfloor) = 1$, then using $(\star)$ again we get that β can be extended to a partial assignment $\alpha \in [r]^{B(T_t)}$ that satisfies every constraint C_e^ϕ with $e \subseteq B_u$ for some $u \in T_v$. However, $B(T_t) = B(T) = V$ by property (ii) of Definition 3.4.8 and the assumption that $\mathcal{H}$ is connected. Furthermore, α satisfies every constraint of ϕ, since by the same property (ii) for any $e \in E$ there exists $u \in T$ for which $e \subseteq B_u$. In other words, ϕ is satisfiable, which proves our claim.

During the rest of the proof, we will also view each block of gates G_v as a family of assignments in the sense of Definition 3.4.1. We will present the proof of the hardest case of the induction step. The rest of the argument follows using similar ideas. Assume that $(\star)$ holds for every node $u \in T$ with $\mathsf{depth}(u) < i$. We prove the induction step for all $v \in T$ of depth i by constructing (in parallel) appropriate gate families $G_v = \{g_\beta^v\}_{\beta \in [r]^{B_v}}$. The most interesting case of the induction step happens when $|S(v)| = 2$, i.e., v has successors $u_1, u_2 \in T$. Observe that $\mathsf{depth}(u_1) < i$ and $\mathsf{depth}(u_2) < i$, thus we can apply the induction hypothesis to these nodes. We claim that in order to satisfy $(\star)$ it is enough to set

$$G_v \stackrel{\text{def}}{=} \pi_{B_v}\Big(\big(\underbrace{[r]^{B_v} \bowtie \big(\bowtie_{e\in E, e\subseteq B_v} C_e\big)}_{C_1}\big) \bowtie \big(\underbrace{\bowtie_{u\in S(v)} G_u}_{C_2}\big)\Big), \tag{3.8}$$

where C_e is the block of input variables of the circuit corresponding to the constraint associated to hyperedge e. For convenience, let $C_1 \stackrel{\text{def}}{=} [r]^{B_v} \bowtie \big(\bowtie_{e\in E, e\subseteq B_v} C_e\big)$ and $C_2 \stackrel{\text{def}}{=} \bowtie_{u\in S(v)} G_u$. Observe that, for any fixed input $\lceil\phi\rfloor$, $C_1^\phi \subseteq [r]^{B_v}$ and $C_2^\phi \subseteq [r]^{B_{u_1}\cup B_{u_2}}$.

In order to prove our claim, fix $\beta \in [r]^{B_v}$. First, assume that β can be extended to a partial assignment $\alpha \in [r]^{B(T_v)}$ satisfying every constraint C_e^ϕ with $e \subseteq B_u$ for some $u \in T_v$. Then $\alpha|_e \in C_e^\phi$ for each $e \subseteq B_v$, which implies that $\alpha|_{B_v} \in C_1^\phi$. In addition, it follows from the definition of α and the induction hypothesis that, under $\lceil\phi\rfloor$, we have $\alpha|_{B_u} \in G_u^\phi$, for each $u \in S(v)$. But then $\alpha|_{B_v\cup B_{u_1}\cup B_{u_2}} \in C_1^\phi \bowtie C_2^\phi$, and since α extends β, we get $\beta = \alpha|_{B_v} \in \pi_{B_v}(C_1^\phi \bowtie C_2^\phi)$. Therefore, $G_v^\phi(\beta) = 1$, which completes one direction of the argument.

Now assume that, for a fixed input $\lceil\phi\rfloor$, we have $G_v^\phi(\beta) = 1$. We need to show how to extend β to a partial assignment $\alpha \in [r]^{B(T_v)}$ satisfying the condition in $(\star)$. First, notice that β already satisfies every constraint C_e^ϕ with $e \subseteq B_v$ according to our definition in Equation (3.8). Further, using (3.8) once again and the induction hypothesis, there exist

$\beta_1 \in [r]^{B_{u_1}}$ and $\beta_2 \in [r]^{B_{u_2}}$ that agree with β over $B_{u_1} \cap B_v$ and $B_{u_2} \cap B_v$, respectively, and that can be extended to partial assignments $\alpha_1 \in [r]^{B(T_{u_1})}$ and $\alpha_2 \in [r]^{B(T_{u_2})}$ satisfying the condition in $(\star)$ with respect to T_{u_1} and T_{u_2}, respectively. It follows then from Lemma 3.4.9 that $\alpha \stackrel{\text{def}}{=} \beta \cup \alpha_1 \cup \alpha_2 \in [r]^{B(T_v)}$ is a *well-defined function* which satisfies every constraint C_e^{ϕ} with $e \subseteq B_u$ for some $u \in T_v$. This completes the proof of our claim.

It remains to argue that M can be implemented without negations, and to upper bound the depth of this circuit. First, the final step of the computation as defined in Equation (3.7) can be performed by fan-in two monotone circuits of depth $O(w \cdot \log r)$. During each core stage $i \in [d]$, we implement in parallel each block of gates G_v as described in Equation (3.8), for all v for which $\mathsf{depth}(v) = i - 1$. The computation of each G_v relies on the computation of the corresponding functions C_1 and C_2. Let's consider the depth increase during the i-stage. According to Lemma 3.4.3, we can compute each output bit of C_2 in monotone depth 1. Similarly, using a balanced binary tree and Lemma 3.4.3, we can compute each output bit of C_1 by fan-in two monotone circuits of depth $O(\log \binom{w}{k}) = O(k \cdot \log w)$. Given the computations of C_1 and C_2, each gate in G_v can now be computed via Proposition 3.4.7 in monotone depth $O(w \cdot \log r)$. Since there are d stages, the monotone depth of M can be upper bounded by:

$$\begin{aligned} \mathsf{depth}^{+}(M) &= O(d \cdot (k \cdot \log w + w \cdot \log r)) \\ &= O_{r,k}(d \cdot w) \\ &= O_{r,k}(\mathsf{depth}(\mathcal{D}) \cdot \mathsf{width}(\mathcal{D})) \\ &= O_{r,k}(\mathsf{depth\text{-}width}(\mathcal{H})), \end{aligned}$$

which completes the proof of our result. □

3.5 An unconditional classification theorem for CSPs

This section contains the proof of Theorem 3.1.13. It relies on several techniques introduced in earlier sections, together with a few additional results. We start with the following connection between the tree-width of a hypergraph $\mathcal{H}$, and its depth-width complexity.

Proposition 3.5.1 (Bodlaender [34])**.** *Let $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be a sequence of k-uniform hypergraphs, where each $\mathcal{H}_n$ has n vertices, and $k \in \mathbb{N}$. Then,*

$$\mathsf{tree\text{-}width}(\mathcal{H}_n) \;\leq\; \mathsf{depth\text{-}width}(\mathcal{H}_n) \;\leq\; O(\log n \cdot \mathsf{tree\text{-}width}(\mathcal{H}_n)).$$

Proof. This result follows from [Proposition 11.27, 67], which relates the tree-width of a hypergraph $\mathcal{H}$ to the tree-width of its Gaifman graph G, and [Theorem 4.2, 34], which shows that if G is a graph on n vertices that admits a tree-decomposition of width ℓ, then it also admits a balanced decomposition of width $O(\ell)$ and depth $O(\log n)$. □

The next lemma shows that if a constant-degree graph is not too far from a tree, then its Tseitin hypergraph satisfies the same property.

Lemma 3.5.2. *Let $G = (V_G, E_G)$ be a k-regular undirected graph on n vertices, and $\mathcal{H} = \mathsf{Tseitin}(G)$. Then,*

$$\mathsf{tree\text{-}width}(\mathcal{H}) \;\leq\; O_k(\mathsf{tree\text{-}width}(G)).$$

Proof. Let $\mathcal{D}_G = (\mathcal{T}_G, \{B_t\}_{t\in T_G})$ be a tree decomposition of G of width w, where $\mathcal{T}_G = (T_G, F_G)$. Let $\mathcal{H} = \mathsf{Tseitin}(G)$, and recall that:

$$V_{\mathcal{H}} \stackrel{\text{def}}{=} \{v_e \mid e = \{u, w\} \text{ is an edge of } G\} \quad \text{and} \quad E_{\mathcal{H}} \stackrel{\text{def}}{=} \{e_u \mid u \in V_G \text{ and } e_u = {\downarrow}(N_G(u))\},$$

where ${\downarrow}(N_G(u)) \stackrel{\text{def}}{=} \{v_{\{u,w\}} \mid w \in N_G(u)\}$. We construct a decomposition $\mathcal{D}_{\mathcal{H}} = (\mathcal{T}_{\mathcal{H}}, \{B_t\}_{t\in T_{\mathcal{H}}})$ of $\mathcal{H}$ as follows, where $\mathcal{T}_{\mathcal{H}} = (T_{\mathcal{H}}, F_{\mathcal{H}})$. First, we let $\mathcal{T}_{\mathcal{H}}$ be isomorphic to $\mathcal{T}_G$. For convenience, for every $t_G \in T_G$, we use $t_{\mathcal{H}}$ to denote the corresponding node of $T_{\mathcal{H}}$. The crucial distinction occurs in the definition of the bags of $\mathcal{D}_{\mathcal{H}}$. For every $t_{\mathcal{H}} \in T_{\mathcal{H}}$, we let

$$B_{t_{\mathcal{H}}} \stackrel{\text{def}}{=} \bigcup_{u\in B_{t_G}} \{v_e \mid w \in V(G) \text{ and } e = \{u, w\} \in E(G)\}. \tag{3.9}$$

Since G is k-regular,

$$\mathsf{width}(\mathcal{D}_{\mathcal{H}}) \;\leq\; \max_{t_{\mathcal{H}}\in T_{\mathcal{H}}} |B_{t_{\mathcal{H}}}| \;\leq\; \max_{t_{\mathcal{H}}\in T_{\mathcal{H}}} k \cdot |B_{t_G}| \;\leq\; k \cdot (w+1) \;\leq\; O_k(w).$$

We argue next that this is indeed a tree decomposition of $\mathcal{H}$. First, since every vertex of G is in at least one bag of $\mathcal{D}_G$, we get from Equation (3.9) that every hyperedge of $\mathcal{H}$ is contained in some bag of $\mathcal{D}_{\mathcal{H}}$. Similarly, every vertex v_e of $\mathcal{H}$ is in some bag of $\mathcal{D}_{\mathcal{H}}$,

and the set $B^{-1}_{\mathcal{D}_\mathcal{H}}(v_e)$ is nonempty. Finally, fix a vertex $v_e \in V_\mathcal{H}$, where $e = \{u, w\}$ is an edge of G. Consider again the set $B^{-1}_{\mathcal{D}_\mathcal{H}}(v_e) \subseteq T_\mathcal{H}$, and recall that this set contains a node $t_\mathcal{H} \in T_\mathcal{H}$ if and only if $v_e \in B_{t_\mathcal{H}}$. We need to prove that $\mathcal{T}_\mathcal{H}[B^{-1}_{\mathcal{D}_\mathcal{H}}(v_e)]$ is connected, or more precisely, that the underlying undirected tree is connected. Since v_e corresponds to the edge $e = \{u, w\}$ in G, it is not hard to see that, according to our definition in Equation (3.9),

$$t_\mathcal{H} \in B^{-1}_{\mathcal{D}_\mathcal{H}}(v_e) \subseteq T_\mathcal{H} \quad \iff \quad t_G \in B^{-1}_{\mathcal{D}_G}(u) \cup B^{-1}_{\mathcal{D}_G}(w) \subseteq T_G. \tag{3.10}$$

Since $\mathcal{D}_G$ is a decomposition of G, we know that both $\mathcal{T}_G[B^{-1}_{\mathcal{D}_G}(u)]$ and $\mathcal{T}_G[B^{-1}_{\mathcal{D}_G}(w)]$ are connected subgraphs of $\mathcal{T}_G$. Therefore, the corresponding sets of nodes of $\mathcal{T}_\mathcal{H}$ are also connected, since $\mathcal{T}_\mathcal{H} \cong \mathcal{T}_G$ by definition. Finally, using again the definition of tree decomposition, there exists a node $t_G \in T_G$ such that $e = \{u, w\} \subseteq B_{t_G}$ in $\mathcal{D}_G$, as e is an edge of G. But then $B^{-1}_{\mathcal{D}_G}(u) \cap B^{-1}_{\mathcal{D}_G}(w)$ is nonempty, $\mathcal{T}_G[B^{-1}_{\mathcal{D}_G}(u) \cup B^{-1}_{\mathcal{D}_G}(w)]$ is connected, and from Equation (3.10) and the isomorphism between the trees, $\mathcal{T}_\mathcal{H}[B^{-1}_{\mathcal{D}_\mathcal{H}}(v_e)]$ must be connected. □

Recall that an undirected graph H is called a *minor* of a graph G if an isomorphic copy of H can be obtained from G by deleting edges and vertices and by contracting edges. An equivalent definition is that there exists a function $\zeta\colon V(H) \to \mathcal{P}(V(G)) \setminus \emptyset$ that maps vertices of H to non-empty subsets of $V(G)$ for which the following holds:

(*i*) For every $v \in V(H)$, $G[\zeta(v)]$ is connected;

(*ii*) For distinct vertices $u, v \in V(H)$, we have $\zeta(u) \cap \zeta(v) = \emptyset$;

(*iii*) If $\{u, v\} \in E(H)$, then there exists an edge connecting a vertex in $\zeta(u)$ to a vertex in $\zeta(v)$.

If this is the case, we say that ζ is an *embedding* of H in G.

Recall that an undirected graph H on $\ell_1 \cdot \ell_2$ vertices is an $\ell_1 \times \ell_2$-*grid* if H is isomorphic to the graph Cartesian product of P_{ℓ_1} and P_{ℓ_2}, where P_ℓ denotes the *path* on ℓ vertices. The next lemma formalizes the intuitive fact that graphs containing a large grid as a minor have good routing properties.

Lemma 3.5.3. *There exists a real-valued constant $\delta > 0$ for which the following holds. Let G be an undirected graph, and suppose that G contains an $\ell \times \ell$-grid as a minor. Then* $\mathsf{routing}(G) \geq \ell^\delta$.

Proof. Let H be an $\ell \times \ell$-grid, and ζ be an embedding of H in G. It is possible to show that H contains a set of vertices $W_H \subseteq V(H)$ of size $2\lceil \ell^\delta \rceil$, where $\delta > 0$ is independent of ℓ, for which the following holds: for any set of $\lceil \ell^\delta \rceil$ disjoint pairs of vertices in W_H, there exist *vertex-disjoint* paths in H that connect these pairs of vertices. This can be seen by considering vertices on a horizontal line that are sufficiently spaced, and some canonical way of constructing vertex-disjoint paths between the vertices (a closely related problem is investigated by Cutler and Shiloach [53] and Aggarwal et al. [5]).

Now for each $v \in W_H$, let $v' \in V(G)$ be a representative for v in $\zeta(v)$, and $W_G \subseteq V(G)$ be the corresponding set of vertices in G. Then, since ζ is an embedding of H in G, it follows that for any set of disjoint pairs of vertices in W_G, there exist *edge-disjoint* paths in G that connect these pairs of vertices.[7] In other words, $\mathsf{routing}(G) \geq \ell^\delta$. □

Finally, we will need the following strengthening of the Grid-Minor Theorem (Robertson and Seymour [159]).

Proposition 3.5.4 (Chekuri-Chuzhoy [48])**.** *There exists a real-valued constant $\delta > 0$ for which the following holds. If G is an undirected graph with tree-width t, then G contains an $\ell \times \ell$-grid as a minor, where $\ell \geq t^\delta$.*

We are ready to give the proof of Theorem 3.1.13, stated again below.

Theorem. *There exists a fixed constant $c > 0$ for which the following holds. If $\{G_n\}_{n\in\mathbb{N}}$ is a sequence of undirected graphs, where each G_n is a k-regular connected graph on n vertices, then*

$$\Omega(\mathsf{tree\text{-}width}(G_n)^c) \;\leq\; \mathsf{depth}^+\big(\mathsf{CSP}_{\mathsf{Tseitin}(G_n),3}\big) \;\leq\; O_k(\mathsf{tree\text{-}width}(G_n) \cdot \log n).$$

Proof. Let $\{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be the sequence of k-uniform Tseitin hypergraphs associated to $\{G_n\}_{n\in\mathbb{N}}$, i.e., each $\mathcal{H}_n \stackrel{\text{def}}{=} \mathsf{Tseitin}(G_n)$. Assume for convenience that each G_n has tree-width $\gamma(n)$, where $\gamma\colon \mathbb{N} \to \mathbb{N}$.

[7] Note that it is not clear whether edge-disjoint paths in the grid can be kept edge-disjoint when we embed the grid in another graph G (consider, for instance, an internal vertex of the grid that becomes an edge in G). This is the reason why we start with vertex-disjoint paths.

We start with the upper bound. We know from Lemma 3.5.2 that each $\mathcal{H}_n$ has tree-width $O_k(\gamma(n))$. As a consequence, we get from Proposition 3.5.1 that each $\mathcal{H}_n$ has depth-width $O_k(\gamma(n) \cdot \log n)$. The upper bound now follows immediately from Theorem 3.1.11.

For the lower bound, it follows from Proposition 3.5.4 that each G_n contains an $\ell(n) \times \ell(n)$-grid as a minor, where $\ell(n) \geq \gamma(n)^\delta$, and $\delta > 0$ is a fixed constant. Consequently, Lemma 3.5.3 implies that G_n has routing number at least $\gamma(n)^c$, for a fixed constant $c > 0$. Now fix a coloring function $\chi_n \colon V(G_n) \to \{0, 1\}$ that forces the CSP $\phi_{G_n}^{\chi_n}$ to be unsatisfiable. Recall that this is possible according to Fact 3.3.3. Moreover, each CSP $\phi_{G_n}^{\chi_n}$ has geometry $\mathcal{H}_n$. Using Proposition 3.3.5 and the definition of sensitivity for hypergraphs (Definition 3.1.9), it follows that $\mathsf{sens}(\mathcal{H}_n) \geq \gamma(n)^c$. The lower bound follows immediately from Corollary 3.1.10, which completes the proof. □

3.6 Example: The depth-width of the Cycle

We say that an undirected graph $G_n = (V_n, E_n)$ is the *cycle* on n vertices if $V_n = \{v_1, \ldots, v_n\}$ and $E_n = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}, \{v_1, v_n\}\}$. The *path* graph on n vertices is obtained from this graph by deleting the edge $\{v_1, v_n\}$. We describe next an essentially optimal tree decomposition of C_n with respect to depth-width complexity.

Proposition 3.6.1. *If C_n denotes the cycle on n vertices, then*

$$\mathsf{depth\text{-}width}(C_n) = \Theta(\log n).$$

Proof. For the lower bound, observe that Theorem 3.1.11 implies that

$$\mathsf{depth\text{-}width}(C_n) = \Omega(\mathsf{depth}^+(\mathsf{CSP}_{C_n,2})). \tag{3.11}$$

Since $\mathsf{CSP}_{C_n,2}$ is a Boolean function that depends on all of its $4n$ input variables, and we consider fan-in two circuits, it must be the case that $\mathsf{depth}^+(\mathsf{CSP}_{C_n,2}) = \Omega(\log n)$. This fact and inequality (3.11) establish the lower bound.

We now describe an explicit tree decomposition for C_n. Disregarding constant factors, it is enough to obtain a tree decomposition $\mathcal{D}_{P_n} = (\mathcal{T}, \{B_t\}_{t \in T})$ for the path graph $P_n = (V_n, E_n)$, where $\mathcal{T} = (T, A)$ (simply add v_1 and v_n to each bag B_t of $\mathcal{D}_{P_n}$ in order to

obtain a decomposition for C_n). For simplicity, assume that $n = 2^\ell + 1$, and recall that $V_n = \{v_1, \ldots, v_n\}$. The general case can be handled by an easy extension of the construction described below.

We let $\mathcal{T}$ be a complete (rooted, directed) binary tree with ℓ layers. The leaves of $\mathcal{T}$ correspond to nodes $t_1, \ldots, t_{2^\ell}$. The remaining nodes of T are named arbitrarily. First, let $B_{t_i} \stackrel{\text{def}}{=} \{v_i, v_{i+1}\}$. Observe that, except for v_1 and v_n, each vertex of P_n occurs in exactly two leaves of $\mathcal{T}$. In order to define the bag B_w of a non-leaf node $w \in T$, consider the following process. For every vertex $v_i \in V_n$ different from v_1 and v_n, consider the path $P' \subseteq A$ that connects the two leaves containing v_i, and add v_i to the bag B_w of each non-leaf node $w \in T$ that occurs in P'.[8] This completes the description of $\mathcal{D}_{P_n}$.

We argue next that $\mathcal{D}_{P_n}$ is a tree-decomposition of P_n. First, observe that, by construction, the second condition in the definition of tree decomposition (Definition 3.4.8) is satisfied. In addition, the first condition is satisfied as well, based on our construction for the bags of the non-leaf nodes of $\mathcal{T}$, and the definition of the bag of each leaf node.

Finally, we show that $\mathsf{depth\text{-}width}(\mathcal{D}_{P_n}) = O(\log n)$. Indeed, we have $\mathsf{depth}(\mathcal{T}) = \ell = O(\log n)$, and in order to complete the argument we prove that $\mathsf{width}(\mathcal{D}_{P_n}) \leq 3$. This can be seen as follows. First, for every leaf node $t_i \in T$, we have $|B_{t_i}| = 2$. On the other hand, let $w \in T$ be a non-leaf node of $\mathcal{T}$, and consider the complete subtrees $\mathcal{T}_{\mathsf{left}}, \mathcal{T}_{\mathsf{right}} \subseteq \mathcal{T}$ to the left and to the right of the parent node w. Let u_-^{left} and u_+^{left} be the leaves t_i and t_j in $\mathcal{T}_{\mathsf{left}}$ with the smallest and largest indexes i and j, respectively. We use u_-^{right} and u_+^{right} for the analogue nodes in $\mathcal{T}_{\mathsf{right}}$. Finally, consider the bag of each such node,

$$B_{u_-^{\mathsf{left}}} = \{v_k, v_{k+1}\},\ B_{u_+^{\mathsf{left}}} = \{v_{k'}, v_{k'+1}\},\ B_{u_-^{\mathsf{right}}} = \{v_{k'+1}, v_{k'+2}\},\ \text{and}\ B_{u_+^{\mathsf{right}}} = \{v_{k''}, v_{k''+1}\},$$

where $k < k' < k''$ are appropriate indexes. We claim that there are at most 3 paths passing through w according to the definition of the bags of $\mathcal{D}_{P_n}$. First, there is the path connecting u_+^{left} to u_-^{right}, since $v_{k'+1} \in B_{u_+^{\mathsf{left}}} \cap B_{u_-^{\mathsf{right}}}$. There may be a path connecting u_-^{left} to the immediate leaf to the left of u_-^{left}, since the corresponding bags will both contain v_k (the vacuous case occurs when $v_k = v_1$). Similarly, there may be a path leaving u_+^{right} that ends at its immediate successor leaf. The other paths used during the definition of the bags

[8] Recall that in a tree every pair of distinct vertices is connected by a unique path.

of the tree decomposition are either internal to $\mathcal{T}_{\mathsf{left}}$ or $\mathcal{T}_{\mathsf{right}}$, or do not enter the subtree of $\mathcal{T}$ rooted at w. Since at most 3 paths pass trough w, its bag has size at most 3, which completes the proof. □

Chapter 4

Majority is incompressible by $\mathsf{AC}^0[p]$ circuits

4.1 Background, results, and organization

Computational complexity theory investigates the complexity of solving explicit problems in various computational models. While fairly strong lower bounds are known for restricted models such as constant-depth circuits (Ajtai [8], Furst, Saxe, and Sipser [75], Yao [205], and Håstad [95]) and monotone circuits (Razborov [155], Andreev [16], and Alon and Boppana [12]), our understanding of general Boolean circuits is still very limited. For example, our current state of knowledge does not rule out that every function in $\mathsf{NTIME}(2^n)$ is computed by Boolean circuits of linear size.

Several barriers have been identified to proving lower bounds for general Boolean circuits, such as relativization (Baker, Gill, and Solovay [24]), algebrization (Aaronson and Wigderson [2]), and the "natural proofs" barrier (Razborov and Rudich [154]). Most known lower bound techniques for restricted models are "naturalizable", and it is believed that substantially different methods will be required in order to prove strong lower bounds for unrestricted models.

In spite of this, the techniques used to prove lower bounds for weaker models are still interesting, and an improved understanding of these techniques can have substantial benefits. First, there is a developing theory of connections between unconditional lower bounds

and algorithmic results, which involves satisfiability algorithms, learning algorithms, truth-table generation, among other models (cf. Williams [202], Oliveira [148], and Santhanam [168]). In particular, such connections provide new insights and results in both areas, and a better understanding of restricted classes of circuits can lead to improved algorithms (cf. Williams [203]). Second, strong enough lower bounds for weaker models imply lower bounds for more general models (Valiant [190, 192], see Viola [195] for a modern exposition). In a similar vein, we mention the surprising results from Allender and Koucký [10] showing that, in some cases, weak circuit size lower bounds of the form $n^{1+\varepsilon}$ yield much stronger results.

Furthermore, even if known proof techniques individually naturalize, it is possible they could be used as ingredients of a more sophisticated approach which is more powerful. A recent striking example of this is the use by Williams [204] of structural characterizations of ACC^0 circuits, together with various complexity tools such as completeness for problems on succinctly represented inputs, diagonalization, and the easy witness method, in order to separate NEXP from ACC^0. Given the paucity of techniques in the area of complexity lower bounds, it makes sense to try to properly understand the techniques we do have.

We focus in this chapter on $\mathcal{C}$-compression games (Chattopadhyay and Santhanam [45]), where $\mathcal{C}$ is some class of Boolean circuits. A $\mathcal{C}$-compression game is a 2-player (interactive) communication game where the first player Alice is computationally bounded (by being restricted to play strategies in $\mathcal{C}$) and has access to the entire input $x \in \{0,1\}^n$, while the second player Bob is computationally unbounded and initially has no information about the input. Alice and Bob communicate to compute $f(x)$ for a fixed Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, and the question is how many bits of communication sent by Alice are required. Note that if f is computable by $\mathcal{C}$, then 1 bit of communication suffices, as Alice can compute $f(x)$ by herself, and send the answer to Bob. Thus, if we are interested in unconditional lower bounds on the communication cost for an explicit function, we must study circuit classes $\mathcal{C}$ where lower bounds are already known for explicit functions, such as constant-depth circuits, and their extension with modulo p gates.

Compression games hybridize between communication complexity and computational complexity as follows. In the traditional two-party communication complexity model, Alice and Bob are symmetric – they each know half of the input, and communicate to compute

a given function on the whole input. Neither party is computationally bounded. Thus they are equally constrained (or unconstrained) informationally as well as computationally. In the compression game setting, an asymmetry appears. Alice now has an informational advantage over Bob – she begins with knowledge of the whole input, while Bob has no knowledge about the input at all. However, this informational advantage is offset by a computational constraint – Alice can only use strategies computable from $\mathcal{C}$. Thus studying compression games can be thought of as studying the tradeoff between information and computation. Typically, when studying the question of lower bounds against $\mathcal{C}$, we are merely interested in whether a function f is computable in $\mathcal{C}$ or not. Now, we are concerned instead by how much information can be obtained about $f(x)$ using merely circuits from $\mathcal{C}$, or conversely, how much assistance a $\mathcal{C}$-bounded party requires from an unbounded one in order to compute $f(x)$. In other terms, we would like to obtain a refined quantitative picture of solvability by $\mathcal{C}$-circuits, rather than a purely qualitative one.

Communication complexity has long been an important tool in the complexity theorist's toolkit. In particular, several lower bound techniques such as the crossing sequence method, the Nečiporuk method [143] and the Khrapchenko method [119] can be interpreted as uses of communication complexity (cf. Kushilevitz and Nisan [128]). Often, when a computational model is relatively weak, lower bound techniques exploit some sort of information bottleneck in the model, which is how communication complexity enters the picture. By studying compression games, where the model explicitly incorporates both communication and computation, we hope to better understand the interplay between communication complexity techniques and computational complexity techniques.

We explore in this chapter the power of the *polynomial approximation method* (Razborov [157], Smolensky [177]) and the *random restriction method* (cf. Furst, Saxe, and Sipser [75] and Håstad [95]) in the context of interactive compression games. We use these techniques and the compression framework to prove significant generalizations of known lower bounds for constant-depth circuits.

Compression games have been considered before, both to prove unconditional and conditional lower bounds. The pioneering work of Dubrov and Ishai [60] showed that Parity_n requires AC^0-compression cost $n^{1-\varepsilon}$ (for any fixed $\varepsilon > 0$, and large enough n) when there

is only one round of communication between Alice and Bob. Dubrov and Ishai were motivated by questions about the randomness complexity of sampling, and their work has later found applications in leakage-resilient cryptography (Faust et al. [61]). Chattopadyay and Santhanam [45] strengthened the Dubrov-Ishai lower bound to $n/\mathsf{poly}(\log n)$, and showed that the lower bound holds for multi-round games where Alice is allowed to use a randomized strategy. Their main technique was a generic connection between correlation and multi-round compression. As strong correlation lower bounds are not known for $\mathsf{AC}^0[p]$ circuits (see e.g. Srinivasan [179]), their technique does not yield strong lower bounds for multi-round $\mathsf{AC}^0[p]$-compression games, which constitute the main topic of this chapter.

The investigation of single-round compression (also known as instance compression) has found connections to other topics in areas such as cryptography (Harnik and Naor [93]), parameterized complexity (cf. Bodlaender et al. [33]), probabilistic checkable proofs (Fortnow and Santhanam [71]), and structural complexity (Buhrman and Hitchcock [43]), and has received considerable attention recently (see e.g. Drucker [59] and Dell [57]). There has also been a long line of work on proving lower bounds for $\mathsf{SIZE}(\mathsf{poly}(n))$-compression games under complexity-theoretic assumptions (cf. Dell and van Melkebeek [56]), but papers along this line use very different ideas, and hence are tangential to our work.

For a circuit class $\mathcal{C}$, we use $\mathcal{C}_d$ to denote the restriction of $\mathcal{C}$ to polynomial size circuits of depth d. For instance, AC^0_d refers to polynomial size depth-d circuits. Recall that $\mathsf{Majority}_n \colon \{0,1\}^n \to \{0,1\}$ is the function that is 1 on an input x if and only if $\sum_{i\in[n]} x_i \geq n/2$. Further, we let $\mathsf{MOD}^n_q \colon \{0,1\}^n \to \{0,1\}$ be the function that is 1 if and only if q divides $\sum_{i\in[n]} x_i$.

The proof that $\mathsf{Majority}_n \notin \mathsf{AC}^0_d[p]$ for $d(n) = o(\log n/\log\log n)$ (Razborov [157], Smolensky [177]) remains one of the strongest lower bounds for an explicit function. There are no known explicit lower bounds for polynomial size circuits of depth $d = \omega(\log n/\log\log n)$, nor for constant depth circuits with arbitrary (composite) modulo gates.

In the framework of compression games, the Razborov-Smolensky lower bound is equivalent to the claim that in any $\mathsf{AC}^0[p]$ game for $\mathsf{Majority}$, there must be non-trivial communication between Alice and Bob. More recently, Chattopadhyay and Santhanam [45] proved that in any *randomized single-round* $\mathsf{AC}^0_d[p]$-compression protocol for this function, Al-

ice must communicate $\sqrt{n}/(\log n)^{O(d)}$ bits. However, their technique does not extend to multiple-round compression games. Before this work, the only known technique to prove unconditional lower bounds for games with an arbitrary number of rounds used a connection between compressibility and correlation. The lack of strong correlation bounds for low-degree $\mathbb{F}_p$ polynomials computing explicit Boolean functions prevents us from using this connection to get $\mathsf{AC}^0[p]$-compression lower bounds (see Srinivasan [179] for more details).

In this work, we bypass this difficulty through a new application of the polynomial approximation method, obtaining the following result.

Theorem 4.1.1. *Let p be a prime number. There exists a constant $c \in \mathbb{N}$ such that, for any $d \in \mathbb{N}$, and every $n \in \mathbb{N}$ sufficiently large, the following holds.*

(i) Any $\mathsf{AC}^0_d[p]$-compression game for $\mathsf{Majority}_n$ (with any number of rounds) has communication cost at least $n/(\log n)^{2d+c}$.

(ii) There exists a single-round AC^0_d-compression game for $\mathsf{Majority}_n$ with communication cost at most $n/(\log n)^{d-c}$.

The argument for the lower bound part of this result proceeds roughly as follows. First, we show via a reduction in the interactive compression framework that a protocol for $\mathsf{Majority}_n$ can be used to compress other symmetric functions, such as MOD^n_q. In other words, it is enough to prove a strong communication lower bound for MOD^n_q in order to establish the lower bound in Theorem 4.1.1. We then employ a general technique that allows us to transform an interactive protocol for a Boolean function f into an exponentially large circuit computing f, following an approach introduced in Chattopadhyay and Santhanam [45]. We have thus reduced the original problem involving computation and communication to a certain circuit lower bound for MOD_q.

A crucial ingredient in our proof is a new exponential lower bound for a certain class of bounded-depth circuits extended with modulo p gates computing the MOD_q function. Although obtaining circuit lower bounds for depth d circuits beyond size roughly $2^{n^{1/(d-1)}}$ is a major open problem in circuit complexity (see e.g. Viola [195]), we show that, under a certain *semantic* constraint on the $\mathsf{AC}^0_d[p]$ circuit, MOD^n_q requires circuits of size $2^{n/(\log n)^{O(d)}}$. More specifically, we consider circuits consisting of a disjunction of exponentially many

polynomial size circuits, for which the following holds: whenever the top gate evaluates to true, precisely one subcircuit evaluates to true.

The proof of this circuit lower bound relies on the application of the polynomial approximation method in the *exponentially small error regime*, as opposed to the original proofs of Razborov and Smolensky, which are optimized with constant error. In particular, this approach allows us to prove a stronger lower bound that avoids the correlation barrier mentioned before. In order to implement this idea, we rely on a recent strengthening of their method introduced by Kopparty and Srinivasan [123], and on an extension of the degree lower bounds of Razborov and Smolensky to very small error. We believe that this new circuit lower bound may be of independent interest, and that semantic restrictions will find more applications in circuit complexity. Altogether, these results give the lower bound in Theorem 4.1.1.

Theorem 4.1.1 implies a new result for $\mathsf{AC}^0[p]$ circuits extended with arbitrary oracle gates, which we state next.

Corollary 4.1.2. *Let $p \geq 2$ be prime, and $d \in \mathbb{N}$. There exists a constant $c \in \mathbb{N}$ such that, for every sufficiently large n, the following holds. If $\mathsf{Majority}_n$ is computed by polynomial-size $\mathsf{AC}^0_d[p]$ circuits with arbitrary oracle gates, then the total fan-in of the oracle gates is at least $n/(\log n)^{2d+c}$.*

Another interesting consequence of Theorem 4.1.1 is that it provides information about the *structure* of polynomial size circuits with modulo p gates computing $\mathsf{Majority}_n$. More precisely, it implies that in any layered circuit, at least $\lfloor n/(\log n)^{2k+c} \rfloor$ gates must be present in the k-th layer, which is essentially optimal.

Observe that Theorem 4.1.1 holds for deterministic compression games. For randomized protocols, in which Alice can employ a probabilistic strategy, we use our techniques to prove the following strengthening over previous results.

Theorem 4.1.3. *Let p and q be distinct primes. There exists a constant $c \in \mathbb{N}$ such that, for any $d \in \mathbb{N}$, and $n \in \mathbb{N}$ sufficiently large, every randomized $\mathsf{AC}^0_d[p]$-compression game for MOD^n_q with any number of rounds and error at most $1/3$ has communication cost at least $\sqrt{n}/(\log n)^{d+c}$.*

We stress that Theorems 4.1.1 and 4.1.3 hold both for $\mathsf{Majority}$ and MOD_q, whenever p and q are distinct primes. Determining the correct communication cost for probabilistic and average-case games for these functions remains an interesting open problem. (We discuss these models in more detail in Section 4.2.)

We also consider a model of *multiparty* compression games. In this framework, Alice is allowed to interact during each round with k additional parties, and the communication cost of the round is defined to be the length of the longest message sent by Alice to one of the parties. Further, the cost of the protocol on a given input is defined as the sum of the costs of the individual rounds. We stress that the extra parties are not allowed to interact with each other during the execution of the protocol.

This is a natural communication framework, motivated by the question of lower bounds for oracle circuits. Lower bounds in this model with a bounded number of rounds imply lower bounds on the maximum individual fan-in of oracle gates in oracle circuits with a bounded number of such layers.

We prove the following bounds on the randomized multiparty $\mathsf{AC}^0[p]$-compression cost of $\mathsf{Majority}$.

Theorem 4.1.4. *Let $p \in \mathbb{N}$ be a fixed prime. For every $k, r, d \in \mathbb{N}$, the following holds.*

(*i*) *There exists a deterministic $n^{1/r}$-party r-round $\mathsf{AC}^0[p]$-compression game for $\mathsf{Majority}_n$ with cost $\widetilde{O}(n^{1/r})$.*

(*ii*) *Every randomized n^k-party r-round $\mathsf{AC}^0_d[p]$-compression game for $\mathsf{Majority}_n$ has cost $\widetilde{\Omega}(n^{1/2r})$.*

The proof of Theorem 4.1.4 also employs the polynomial approximation method, although the argument is different in this case. Observe that this result says that the communication cost of $\mathsf{Majority}_n$ in the randomized multiparty framework is $n^{\Theta(1/r)}$ for r-round protocols. In other words, allowing Alice to interact with more parties for more time reduces communication considerably (under the definition of communication cost for multiparty games).

We obtain a consequence of Theorem 4.1.4 for oracle circuits where there are a bounded number r of such layers, i.e., there are no more than r oracle gates on any input-output

path in the circuit.

Corollary 4.1.5. *Let $p \geq 2$ be prime, and $r, d \in \mathbb{N}$. If $\mathsf{Majority}_n$ is computed by an $\mathsf{AC}^0_d[p]$ circuit of polynomial size with arbitrary oracle gates that contains at most r layers of such gates, then there is some oracle gate with fan-in at least $\widetilde{\Omega}(n^{1/2r})$.*

In fact, lower bounds for multiparty games are connected to the NP versus NC^1 question. It is possible to show that every Boolean function in $\mathsf{NC}^1/\mathsf{poly}$ admits $\mathsf{poly}(n)$-party r-round AC^0-compression games with cost $n^{O(1/r)}$. Thus, proving a lower bound of $n^{\Omega(1)}$ on the cost of $\mathsf{poly}(n)$-party AC^0-compression games with $\omega(1)$ rounds for a function in NP would separate NP from $\mathsf{NC}^1/\mathsf{poly}$. We conjecture that such a lower bound holds for the Clique function. Note that it is already known that strong enough lower bounds on the size of constant-depth circuits for NP functions implies a separation between NP and NC^1 (cf. Viola [195]). The novelty here is that sufficiently strong results about *polynomial-size* constant depth circuits imply similar separations. Essentially, the computation of logarithmic-depth circuits can be factored into constant-depth and low-communication components, and our multiparty communication game models precisely this mixture of notions.

There is an interesting contrast in the statement of Theorem 4.1.1: while the lower bound holds for protocols with any number of rounds, the upper bound is given by a single-round protocol. It is natural to wonder whether in the compression setting interaction allows Alice to solve more computational problems. We provide a natural example of the power of interaction in our framework in Section 4.6, where we observe that, while the inner product function cannot be computed by polynomial size $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits (Hajnal et al. [91]), there exists an efficient two-party $(\mathsf{MAJ} \circ \mathsf{MAJ})$-compression game for this function.

In a similar direction, a *quantitative* study of the power of interaction in two-party compression games was initiated by Chattopadhyay and Santhanam [45] (with respect to AC^0-compression games). They obtained a quadratic gap in communication when one considers r and $(r-1)$-round protocols for a specific Boolean function. We obtain the following strengthening of their round separation theorem.

Theorem 4.1.6. *Let $r \geq 2$ and $\varepsilon > 0$ be fixed parameters. There is an explicit family of functions $f = \{f_n\}_{n \in \mathbb{N}}$ with the following properties:*

(*i*) *There exists an* $\mathsf{AC}^0_2(n)$*-bounded protocol* Π_n *for* f_n *with* r *rounds and cost* $c(n) \leq n^{\varepsilon}$*, for every* $n \geq n_f$*, where* n_f *is a fixed constant that depends on* f*.*

(*ii*) *Any* $\mathsf{AC}^0(\mathsf{poly}(n))$*-bounded protocol* Π *for* f *with* $r-1$ *rounds has cost* $c(n) \geq n^{1-\varepsilon}$*, for every* $n \geq n_\Pi$*, where* n_Π *is a fixed constant that depends on* Π*.*

Our hard function is based on a pointer jumping problem with a grid structure, while Chattopadhyay and Santhanam uses a tree structure. Similar constructions have been used in other works in communication complexity in the information theoretic setting (Papadimitriou and Sipser [150], and subsequent works), but our analysis needs to take into account computational considerations as well.

The proof of Theorem 4.1.6 relies on a careful application of the *random restriction method*, coupled with a round elimination strategy. Observe that the upper bound is achieved by protocols where Alice's strategy can be implemented by linear-size DNFs, while the communication lower bound holds for polynomial size circuits.

Organization of the Chapter. We define interactive compression games and introduce notation in the next section. In Section 4.3, we give the proof of our main result, deferring the discussion of some auxiliary results to the Appendix. Multiparty compression games are discussed in Section 4.4, followed by applications of our communication lower bounds to circuits with oracle gates in Section 4.5. A natural example for which interactive compression can be easier than computation is presented in Section 4.6. The round separation theorem for AC^0 games is proved in Section 4.7. Finally, we mention a few open problems and research directions in Section 4.8.

4.2 Preliminaries and notation

The results of this chapter are essentially self-contained, but some familiarity with basic notions from complexity theory and communication complexity can be helpful. A good introduction to these areas can be found in [19] and [128], respectively.

Basic definitions. We use $\mathsf{Majority}_n$ to denote the Boolean function over n variables

that is 1 if and only if $\sum_i x_i \geq n/2$. For a prime p, we let MOD_p^n be the Boolean function over n variables that is 1 if and only if p divides $\sum_i x_i$. We let $\mathsf{Parity}_n \stackrel{\text{def}}{=} \neg\mathsf{MOD}_2^n$. A function $h\colon \{0,1\}^n \to \{0,1\}$ is *symmetric* if there exists a function $\phi\colon [n] \to \{0,1\}$ such that $h(x) = \phi(\sum_i x_i)$, for every $x \in \{0,1\}^n$. Clearly, $\mathsf{Majority}_n$ and MOD_p^n are symmetric functions. We say that a Boolean function f ε-approximates a Boolean function g over a distribution $\mathcal{D}$ if $\Pr_{\mathbf{x}\sim\mathcal{D}}[f(\mathbf{x}) \neq g(\mathbf{x})] \leq \varepsilon$. An ε-error *probabilistic* polynomial $\mathbf{Q}(x_1,\ldots,x_n) \in \mathbb{F}_p[x_1,\ldots,x_n]$ for a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is a distribution $\mathcal{E}$ over polynomials such that, for every $x \in \{0,1\}^n$, $Pr_{\mathbf{Q}\sim\mathcal{E}}[f(x) \neq \mathbf{Q}(x)] \leq \varepsilon$. The degree of a probabilistic polynomial is the maximum degree over the polynomials on which $\mathcal{E}$ is supported. We say that functions $f\colon \{0,1\}^n \to \{0,1\}$ and $g\colon \{0,1\}^n \to \{0,1\}$ are *disjoint* if $f^{-1}(1) \cap g^{-1}(1) = \emptyset$. We will use p and q throughout this chapter to denote prime numbers, unless noted otherwise.

Languages and circuit classes. We will use $\mathcal{C}$ to denote a circuit class, such as AC^0 and $\mathsf{AC}^0[p]$. Unless stated otherwise, assume that any circuit class discussed in this chapter contains AND, OR, and NOT gates of unbounded fan-in. Our results hold with more general circuit classes, but we stick with this definition for simplicity. The size of a circuit corresponds to the total number of gates in the circuit. We use $\mathcal{C}_d(s(n))$ to denote the same class restricted to circuits of depth d and size $O(s(n))$. For instance, we abuse notation and write $\mathsf{AC}^0_d[p](\mathsf{poly}(n))$ to denote the set of languages decided by polynomial size circuits of depth at most d consisting of unbounded fan-in AND, OR, NOT and MOD_p gates, for a fixed prime $p \in \mathbb{N}$. As a convention, if we write $\mathcal{C}$ without a depth and size specialization, assume that it consists of constant depth polynomial size circuits with gates from $\mathcal{C}$. As usual, we will identify $\mathcal{C}$ both as a set of languages, and as a class of circuits, depending on the context. Furthermore, if C is a fixed circuit, we may also use C to refer to the Boolean function computed by this circuit. The correct meaning will always be clear in both cases.

Deterministic compression games. Given a circuit class $\mathcal{C}$ and a language L, we define a communication game between two players Alice and Bob. The goal is to decide whether a given string $x \in \{0,1\}^n$ belongs to L. We describe this game informally as follows. Alice knows x, but her computational power is limited to functions computed by circuits from $\mathcal{C}$.

On the other hand, Bob can perform arbitrary computations, but has no information about x during the beginning of the game. The players exchange messages during the execution of the protocol, and at the end should be able to decide whether $x \in L$. The goal is to compute the initial function correctly while minimizing the total number of bits sent by Alice during the game.

Formally, a $\mathcal{C}$-bounded protocol $\Pi_n = \langle C^{(1)}, \ldots, C^{(r)}, f^{(1)}, \ldots, f^{(r-1)}, E_n \rangle$ with $r = r(n)$ rounds consists of a sequence of $\mathcal{C}$-circuits for Alice, a strategy for Bob, given by functions $f^{(1)}, \ldots, f^{(r-1)}$, and a set of accepting transcripts E_n. We associate to every protocol Π_n its signature $\mathsf{signature}(\Pi_n) = (n, s_1, t_1, \ldots, t_{r-1}, s_r)$, which is the sequence corresponding to the input size $n = |x|$ and the length of the messages exchanged by Alice and Bob during the execution of the protocol. For convenience, let $s = \sum_{i \in [r]} s_i$, and $t = \sum_{i \in [r-1]} t_i$. We always have $E_n \subseteq \{0,1\}^{t+s}$. In addition, we let $\mathsf{rounds}(\Pi_n) \stackrel{\text{def}}{=} r$. For every $i \in [r]$,

$$C^{(i)} \colon \{0,1\}^{n + \sum_{j<i}(s_j + t_j)} \to \{0,1\}^{s_i},$$

and for every $i \in [r-1]$,

$$f^{(i)} \colon \{0,1\}^{\sum_{j \le i} s_j} \to \{0,1\}^{t_i}.$$

In other words, before the beginning of the i-th round, Alice has sent messages $a^{(i)}, \ldots, a^{(i-1)}$ of size $s_1, \ldots, s_{i-1}$, respectively, and Bob has replied with messages $b^{(1)}, \ldots, b^{(i-1)}$ of size $t_1, \ldots, t_{i-1}$, respectively. Continuing the interaction, the next message sent by Alice is given by $a^{(i)} \stackrel{\text{def}}{=} C^{(i)}(x, a^{(1)}, b^{(1)}, \ldots, a^{(i-1)}, b^{(i-1)})$. On the other hand, since Bob has unlimited computational power, its message during the i-th round is given simply by $b^{(i)} \stackrel{\text{def}}{=} f^{(i)}(a^{(1)}, \ldots, a^{(i)})$. The transcript of Π_n on $x \in \{0,1\}^n$ is the sequence of messages exchanged by Alice and Bob during the execution of the protocol on x, and will be denoted by $\mathsf{transcript}_{\Pi_n}(x) \stackrel{\text{def}}{=} \langle a^{(1)}, b^{(1)}, \ldots, a^{(r)} \rangle \in \{0,1\}^{s+t}$. We say that Π_n solves the compression game of a function $h_n \colon \{0,1\}^n \to \{0,1\}$ if

$$h(x) = 1 \quad \iff \quad \mathsf{transcript}_{\Pi_n}(x) \in E_n.$$

Finally, we let $\mathsf{cost}(\Pi_n) \stackrel{\text{def}}{=} s$. We stress that the length of the messages sent by Bob does not contribute to the cost of the protocol, and we assume for convenience that the length of these messages are limited by the size of the circuits in $\mathcal{C}$. Observe that a *single-round*

game consists of a protocol Π_n with $\mathsf{signature}(\Pi_n) = (n, s_1)$. Put another way, Alice sends a single message $a^{(1)} \in \{0,1\}^{s_1}$, and a decision is made.

Given a language L and a circuit class $\mathcal{C}$, we say that a sequence of $\mathcal{C}$-bounded protocols $\Pi = \{\Pi_n\}_{n\in\mathbb{N}}$ solves the compression game of L with cost $c(n)$ and $r(n)$ rounds if, for every n, Π_n solves the compression game of L_n, and in addition satisfies $\mathsf{cost}(\Pi_n) \leq c(n)$ and $\mathsf{rounds}(\Pi_n) \leq r(n)$.

Observe that if $L \in \mathcal{C}$ then Alice can compute $L(x)$ by herself, and there is a trivial protocol of cost $c(n) = 1$ for L. On the other hand, for every language L there exists a protocol solving its compression game with cost $c(n) \leq n$, since Alice can simply send her whole input to Bob.

Probabilistic and average-case compression games. The definition presented before captures deterministic games computing a function correctly on every input x. Our framework can be extended naturally to probabilistic and average-case games.

First, in a *probabilistic* $\mathcal{C}$-compression game, Alice is allowed to use randomness when computing her next message, while Bob's strategy remains deterministic. Formally, each circuit $C^{(i)}$ has an additional input of uniformly distributed bits, and different circuits have access to independent bits. Clearly, on any $x \in \{0,1\}^n$, $\mathsf{Transcript}_{\Pi_n}(x)$ is now a random variable distributed over $\{0,1\}^{s+t}$. The other definitions remain the same. We say that Π_n solves the compression game of a function $h_n \colon \{0,1\}^n \to \{0,1\}$ with error probability at most $\gamma(n) \in [0,1]$ if, for every $x \in \{0,1\}^n$,

$$\begin{aligned} h_n(x) = 1 &\implies \Pr_{\Pi_n}[\mathsf{Transcript}_{\Pi_n}(x) \in E_n] \geq 1 - \gamma(n), \text{ and if} \\ h_n(x) = 0 &\implies \Pr_{\Pi_n}[\mathsf{Transcript}_{\Pi_n}(x) \in E_n] \leq \gamma(n). \end{aligned}$$

On the other hand, in a *average-case* $\mathcal{C}$-compression game, we have deterministic games as defined before, but allow a small error during the computation of h_n with respect to the uniform distribution over $\{0,1\}^n$. More precisely, we say that a deterministic protocol Π_n solves the compression game of h_n with error at most $\gamma(n) \in [0,1]$ if

$$\Pr_{x\sim\{0,1\}^n}[h_n(x) = 1 \iff \mathsf{transcript}_{\Pi_n}(x) \in E_n] \geq 1 - \gamma(n).$$

These definitions are extended to languages in the natural way. Since in this work all circuit

classes are non-uniform, any probabilistic protocol for a language L with error at most $\gamma(n)$ can be converted into an average-case protocol with error at most $\gamma(n)$ (simply by fixing the randomness of Alice in order to minimize the error probability over $\{0,1\}^n$).

Interacting with several Bobs. We discuss here a more general family of multi-party compression games that allow Alice to interact with multiple Bobs during a single round of the game. The different Bobs are not allowed to communicate with each other, only with Alice. The definition of round complexity for such games is slightly different than for standard 2-party compression games. The reason is as follows. For 2-party games, we can assume that the game concludes with a message to Bob, as Bob is all-powerful and can determine the result of the protocol from the final message. In the case of multi-party games, this assumption isn't well motivated, as no individual Bob might have access to all the information about the protocol. It makes more sense to say the game for a Boolean function h concludes with Alice computing whether $h(x) = 1$, where x is her input. Thus, a 1-round game will consist of Alice sending messages to the various Bobs, the Bobs responding, and finally Alice computing the answer. This naturally extends to a definition of r-round games.

We will also measure the cost of a protocol somewhat differently. We will again count only the communication from Alice to Bob, but the cost of a protocol will not be the sum of the lengths of all messages sent by Alice. Instead, we will define the cost of a round to be the maximum length of a message sent by Alice to some Bob, and then the cost of the protocol to be the sum of the costs over all rounds. This definition of protocol cost is motivated by the connection of our model with lower bounds on oracle circuits, which we elaborate later. A formal definition is presented below.

Let $\mathcal{C}$ be a circuit class, and $k = k(n), r = r(n)$ be arbitrary functions. A $\mathcal{C}$-bounded $(k+1)$-party protocol

$$\Pi_n^{[k]} = \langle D^{(1,1)}, \ldots, D^{(1,k)}; D^{(2,1)}, \ldots, D^{(2,k)}; \ldots; D^{(r+1,1)}, \\ g^{(1,1)}, \ldots, g^{(r,1)}; g^{(1,2)}, \ldots, g^{(r,2)}; \ldots; g^{(1,k)}, \ldots, g^{(r,k)} \rangle$$

with r rounds consists of a sequence of $\mathcal{C}$-circuits for Alice, and strategies for each Bob_i, given by $g^{(1,i)} \ldots g^{(r,i)}$. We associate to every k-party protocol $\Pi_n^{[k]}$ its signature $\mathsf{signature}(\Pi_n^{[k]}) =$

$(n, s_1, t_1, \dots s_r, t_r)$, where for each $j \in [r], i \in [k]$, s_j is the maximum length of a message sent by Alice to any Bob_i during the j-th round, and t_j is the maximum length of a message sent by any Bob_i to Alice during the j-th round. For every $i \in [r], j \in [k]$, $D^{(i,j)}$ maps the sequence of the input x, all messages sent to Alice before the i-th round and all of Alice's messages before the i-th round to Alice's message in the j-th round to Bob_j. $D^{(r+1,1)}$ maps the sequence of x and all messages sent during the protocol to a single bit. For every $i \in [r], j \in [k]$, $g^{(i,j)}$ maps the sequence of all Alice's messages to Bob_j from the first to the i-th round to Bob_j's message to Alice in the i-th round. We say that $\Pi_n^{[k]}$ solves the compression game for a function h_n on n bits if $D^{(r+1,1)}$ outputs 1 on x if and only if $h_n(x) = 1$.

Finally, we let $\mathsf{cost}(\Pi_n^{[k]}) \stackrel{\text{def}}{=} s$, where $s = \sum_{i \in [r]} s_i$. We assume for convenience that the number of parties is always limited by the size of the circuits used by Alice. These definitions extend to languages, probabilistic games, and average-case games in the natural way.

4.3 The communication cost of $\mathsf{AC}^0[p]$-compression games

We start with a construction of single-round compression games for an arbitrary symmetric function.

Lemma 4.3.1. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be an arbitrary symmetric function. Then, for every $1 \le d(n) \le \log n / \log\log n$, the function f admits a single-round $\mathsf{AC}^0_d(\mathsf{poly}(n))$-compression game with communication*

$$c_d(n) = O\left((d-1)! \cdot n \cdot \left(\frac{\log\log n}{\log n}\right)^{d-1}\right).$$

In particular, for every fixed integer $d \ge 1$, we have $c_d(n) = O\big(n/(\log n)^{(d-1)-o(1)}\big)$.

Proof. Let f be a symmetric function that receives as input an n-bit string $x \in \{0,1\}^{[n]}$. We sketch the construction of depth-d circuits for the corresponding compression games. Observe that any integer $n \in \mathbb{N}$ can be represented with at most $\lceil \log(n+1) \rceil$ bits. For simplicity, we will approximate these values by $\log n$. This will be compensated by the use of asymptotic notation in the final bounds.

Observe that for $d = 1$ the result is obvious, since Alice can simply send x to Bob. For every $d \geq 2$, we design an $\mathsf{AC}^0_d(\mathsf{poly}(n))$ circuit that, on a given input x, outputs $m_d \stackrel{\text{def}}{=} (d-2)! \cdot n \cdot (\log\log n)^{d-2}/(\log n)^{d-1}$ binary strings $a^1_d, \ldots, a^{m_d}_d$ of size $s_d \stackrel{\text{def}}{=} (d-1) \cdot \log\log n$, which together encode the number of 1's in x. More precisely, $|x|_1 = \sum_{i=1}^{m_d} \mathsf{dec}(a^i_d)$, where $\mathsf{dec}(a)$ denotes the integer encoded by the binary string a. Therefore, it is enough that Alice communicates in a single-round at most $m_d \cdot s_d$ bits to Bob, which is then able to compute the original value $f(x)$. This last step relies on the assumption that f is a symmetric function.

First, we give a depth-2 circuit with these properties. Partition the n input bits into $m_2 = n/\log n$ blocks of size $t = \log n$. In other words, let $[n] = B_1 \dot\cup \ldots \dot\cup B_{m_2}$, where $|B_i| = t$. For each block B_i, there exists CNFs $\phi^i_1, \ldots, \phi^i_{\log\log n}$ of size $O(n)$ that compute the string $a^i_2 \in \{0,1\}^{\log\log n} = \{0,1\}^{s_2}$ corresponding to the number of 1's in $x_{B_i} \in \{0,1\}^{B_i}$ (the projection of x to B_i). A small formula of this form exists because the number of input bits is $\log n$. Together with the previous discussion, this completes the proof for $d = 2$.

Now fix an arbitrary $d > 2$. We will construct the corresponding AC^0_d circuit by induction. It will be clear from the description that its final size is a polynomial whose leading exponent does not depend on d. Assume that there is a depth $d-1$ circuit C that outputs m_{d-1} strings $a^1_{d-1}, \ldots, a^{m_{d-1}}_{d-1}$, as described before, on any given input $x \in \{0,1\}^n$. Assume also that its top gates are AND gates. This is without loss of generality, given the argument we use below.

Recall that $a^i_{d-1} \in \{0,1\}^{s_{d-1}}$. We partition these strings into m_d sets, each containing $t \stackrel{\text{def}}{=} m_{d-1}/m_d = \log n/((d-2) \cdot \log\log n) \geq 1$ strings, given our upper bound on d. More precisely, we have $[m_{d-1}] = T_1 \dot\cup \ldots \dot\cup T_{m_d}$, where $|T_i| = t$. For convenience, let $A_i = \{a^j_{d-1} \mid j \in T_i\}$. For any a^j_{d-1}, we have $\mathsf{dec}(a^j_{d-1}) \leq 2^{s_{d-1}} = (\log n)^{d-2}$. Consequently,

$$\sum_{j \in A_i} \mathsf{dec}(a^j_{d-1}) \leq |A_i| \cdot (\log n)^{d-2} = t \cdot (\log n)^{d-2} \leq (\log n)^{d-1}.$$

In particular, this sum can be represented with $s_d = (d-1) \cdot \log\log n$ bits. Observe that the strings in A_i have, together, $t \cdot s_{d-1} = \log n$ bits. Therefore, there exists DNFs $\psi^i_1, \ldots, \psi^i_{s_d}$ of size $O(n)$ that compute the sum of the strings in A_i, which we represent as a string $a^i_d \in \{0,1\}^{s_d}$. Since this is the case for every $i \in [m_d]$, we obtain circuits $\psi^i \circ C$ computing

each string a_d^i. Finally, notice that the top three layers of $\psi_j^i \circ C$ can be collapsed into a depth-2 circuit, which gives us an AC_d^0 circuit for the same function. This completes the proof of Lemma 4.3.1. □

Notice that this upper bound comes from a very restricted class of compression games, as there is no continuing interaction with Bob. A simpler and more efficient construction can be obtained for the MOD_q functions, as for them there is no need to keep track of the exact number of 1s in the original input.

As observed by [45], any compression game for $\mathsf{Majority}_{2n}$ can be used to solve the compression game for Parity_n, with some overhead. In general, the same argument provides the following connection, which implies that in order to prove lower bounds for $\mathsf{Majority}$, it is sufficient to get lower bounds for MOD_q.

Proposition 4.3.2. *Let* $h\colon \{0,1\}^n \to \{0,1\}$ *be an arbitrary symmetric function,* $\mathcal{C}$ *be a circuit class, and* $d \geq 1$*. Assume that the* $\mathcal{C}_d(\mathsf{poly}(n))$*-compression game for* $\mathsf{Majority}_n$ *can be solved with cost* $c(n)$ *in* $r(n)$ *rounds. Then the* $\mathcal{C}_{d+O(1)}(\mathsf{poly}(n))$*-compression game for* h *can be solved with cost* $c_h(n) = O(c(2n) \cdot \log n)$ *in* $r_h(n) = O(r(2n) \cdot \log n)$ *rounds.*

Proof. Let Π_{2n}^{Maj} be a protocol for $\mathsf{Majority}_{2n}$. We sketch the construction of a protocol Π_n^h for h. The idea is to run Π_{2n}^{Maj} about $\log n$ times in order to obtain the hamming weight $|x|_1$ of $x \in \{0,1\}^n$, the input given to Alice in the compression game for h.

In order to achieve this, Alice runs Π_{2n}^{Maj} on appropriate inputs of the form $y = x1^k0^{n-k} \in \{0,1\}^{2n}$, where a different k is used during each stage of Π_n^h. Here a stage is simply a complete execution of Π_{2n}^{Maj}, and Alice performs a binary search with at most $O(\log n)$ stages to obtain $|x|_1$. Although we have defined protocols with an implicit set E of accepting transcripts, observe that with an extra round we can ensure that Bob sends the correct output $\mathsf{Majority}_{2n}(y)$ to Alice.

Finally, it is enough to verify that each string y can be computed by constant-depth polynomial size circuits. However, since there are no more than $O(\log n)$ stages, and since Bob sends one bit at each stage, each string y is a function of at most $O(\log n)$ bits, and can certainly be computed by depth-two polynomial size circuits. □

For our main theorem, we will need the following result, whose proof is discussed in more detail in Section 4.9.

Proposition 4.3.3 ([157, 177], folklore)**.** *Let $p, q \geq 2$ be distinct primes. There exist fixed constants $\zeta > 0$ and $n_0 \in \mathbb{N}$ for which the following holds. For every $n \geq n_0$ and $\varepsilon(n) \in [2^{-n}, 1/10q]$, any polynomial $P \in \mathbb{F}_p[x_1, \ldots, x_n]$ that ε-approximates the MOD_q^n function with respect to the uniform distribution has degree at least $\zeta \cdot \sqrt{n \cdot \log(1/\varepsilon)}$.*

Interestingly, our argument relies on a crucial way on the approximation of Boolean circuits by polynomials with exponentially small error. For convenience of the reader, we include the proof of the next result in Section 4.9.

Proposition 4.3.4 ([157, 177, 123])**.** *Let p be a fixed prime. There exists a constant $\alpha = \alpha(p) \in \mathbb{N}$ such that, for every $\delta \in (0, 1/2)$ and $d(n) \geq 1$, any $\mathsf{AC}^0_d[p](s(n))$ circuit C admits a δ-error probabilistic polynomial $\mathbf{Q}(x_1, \ldots, x_n) \in \mathbb{F}_p[x_1, \ldots, x_n]$ of degree at most $(\alpha \cdot \log s)^{d-1} \cdot \log(1/\delta)$. In particular, it follows that for any distribution $\mathcal{D}$ over $\{0,1\}^n$, C is δ-approximated with respect to $\mathcal{D}$ by a polynomial of degree at most $(\alpha \cdot \log s)^{d-1} \cdot \log(1/\delta)$.*

The next proposition is a minor extension of a result implicit in [45]. It allows us to transform an interactive compression protocol for a function into a certain Boolean circuit that computes the same function.

Proposition 4.3.5. *Let $c \colon \mathbb{N} \to \mathbb{N}$ be a function such that $c(n) \leq n$, $s \colon \mathbb{N} \to \mathbb{N}$ be a function with $s(n) = \Omega(n)$, $\gamma \colon \mathbb{N} \to [0, 1/2)$, L be a language, and $\mathcal{C}$ be a circuit class. If there exists an average-case $\mathcal{C}_d(\mathsf{poly}(n))$-compression game for L with cost $c(n)$ and error probability $\gamma(n)$ with respect to the uniform distribution over $\{0,1\}^n$, then there exist circuits $C_1, \ldots, C_T$ from $\mathcal{C}_{d+O(1)}(\mathsf{poly}(n))$, where $T \leq 2^{c(n)}$, such that*

$$\Pr_{x \sim \{0,1\}^n}[L(x) \neq \bigvee_{i \in [T]} C_i(x)] \leq \gamma(n).$$

Furthermore, these circuits are disjoint: $C_i^{-1}(1) \cap C_j^{-1}(1) = \emptyset$ for every pair $i, j \in [T]$ with $i \neq j$.

Proof. Let $\Pi_n = \langle C^{(1)}, \ldots, C^{(r)}, f^{(1)}, \ldots, f^{(r-1)}, E_n \rangle$ be an average-case protocol for L_n with $r(n)$ rounds and error probability $\gamma(n)$. Observe that Π_n solves the $\mathcal{C}$-compression game of

some function $h_n\colon \{0,1\}^n \to \{0,1\}$, and that h_n is $\gamma(n)$-close to L_n. Recall that Π_n has a signature $\mathsf{signature}(\Pi_n) = (n, s_1, t_1, \ldots, t_{r-1}, s_r)$. For convenience, let $t \stackrel{\text{def}}{=} \sum_{i\in[r-1]} t_i$, and $s \stackrel{\text{def}}{=} c(n) = \sum_{i\in[r]} s_i$.

Given a string $w \in \{0,1\}^{s+t}$, we write $w = (w^{(A,1)}, w^{(B,1)}, \ldots, w^{(B,r-1)}, w^{(A,r)})$ as a concatenation of strings whose sizes respect the signature of Π_n. In other words, $|w^{(A,i)}| = s_i$ and $|w^{(B,j)}| = t_i$, for all $i \in [r]$ and $j \in [r-1]$. We say that w is Alice-consistent on an input x if, for every $i \in [r]$, $w^{(A,i)} = C^{(i)}(x, w^{(A,1)}, w^{(B,1)}, \ldots, w^{(B,i-1)})$. On the other hand, we say that w is Bob-consistent if, for every $j \in [r-1]$, $w^{(B,j)} = f^{(j)}(w^{(A,1)}, \ldots, w^{(A,j-1)})$. Observe that whether a string w is Bob-consistent or not does not depend on x. Let $B_n \subseteq \{0,1\}^{t+s}$ denote the set of Bob-consistent strings. For convenience, set $W_n \stackrel{\text{def}}{=} E_n \cap B_n$.

We claim that $h(x) = 1$ if and only if there exists a string $w \in W_n$ that is Alice-consistent on x. One direction is clear, since if $h(x) = 1$ then $\mathsf{transcript}_{\Pi_n}(x) \in E_n$, and this string is both Bob-consistent and Alice-consistent on x. On the other hand, assume there exists $w \in \{0,1\}^{s+t}$ that is Bob-consistent and Alice-consistent on x. An easy induction on the number of rounds of the protocol shows that $w = \mathsf{transcript}_{\Pi_n}(x)$. Furthermore, if $w \in W_n$ then $w \in E_n$, and it must be the case that $h(x) = 1$, since Π_n is a protocol for h_n. Observe that this argument also shows that if $h(x) = 1$ then there is a unique $w \in W_n$ that serves as a certificate for x.

Notice that there are at most $2^{c(n)}$ Bob-consistent strings. This is because for every string $w^A = (w^{(A,1)}, w^{(A,2)}, \ldots, w^{(A,r)}) \in \{0,1\}^s$, there exists a unique completion of w^A by a string $w \in \{0,1\}^{s+t}$ that is Bob-consistent. In particular, $|W_n| \leq 2^{c(n)}$.

For every fixed $w \in W_n$, we claim that there exists a circuit $C_w(x)$ from $\mathcal{C}_{d+O(1)}(\mathsf{poly}(n))$ that checks if w is Alice-consistent on x. Recall that for every $i \in [r]$, $C^{(i)}$ is a circuit from $\mathcal{C}_d(\mathsf{poly}(n))$. Therefore, we can check in parallel whether

$$w^{(A,i)} = C^{(i)}(w^{(A,1)}, w^{(B,1)}, \ldots, w^{(B,i-1)}),$$

for all $i \in [r]$, using just a constant number of additional layers, since we assume throughout that $\mathcal{C}$ has unbounded fan-in AND and OR gates. which proves the claim. It follows that

$$h(x) = \bigvee_{w\in W_n} C_w(x),$$

for every $x \in \{0,1\}^n$. In addition, C_{w_1} and C_{w_2} are disjoint whenever $w_1 \neq w_2$, since exactly one $w \in W_n$ is Alice-consistent on x. Finally, recall that h_n is $\gamma(n)$-close to L_n, which completes the proof of Proposition 4.3.5. □

Proposition 4.3.5 implies that in order to prove communication lower bounds for interactive compression games, it is enough to prove circuit lower bounds of a particular form. We obtain the following result.

Lemma 4.3.6. *Let p and q be distinct primes, $\gamma\colon \mathbb{N} \to (0,1)$ be an arbitrary function, $k \in \mathbb{N}$, and $d = d(n) \in \mathbb{N}$. Assume that*

$$\Pr_{x \sim \{0,1\}^n}[\mathsf{MOD}_q^n(x) \neq \bigvee_{i \in [T(n)]} C_i(x)] \leq \gamma(n),$$

where each C_i is computed by an $\mathsf{AC}^0_d[p](n^k)$ circuit, and C_i and C_j are disjoint whenever $i \neq j$. Then, the following holds.

(i) $\log T(n) \geq \sqrt{n}/(\log n)^{d+O(1)}$ *if* $\gamma(n) \leq 1/20q$;

(ii) $\log T(n) \geq n/(\log n)^{2d+O(1)}$ *in the case of an exact compression game (i.e., $\gamma = 0$).*

Proof. We employ the polynomial approximation method, i.e., we show that if MOD_q^n admits a circuit of this form, then it can be approximated by a polynomial Q whose degree is upper bounded by a function depending on T. We then invoke Proposition 4.3.3 in order to obtain a lower bound on T. More details follow.

First, Proposition 4.3.4 guarantees that for any $\delta > 0$, each circuit C_i can be δ-approximated under the uniform distribution by a polynomial $Q_i \in \mathbb{F}_p[x_1, \ldots, x_n]$ of degree at most $(\ell \cdot \log n)^d \cdot \log(1/\delta)$, where ℓ is a fixed positive constant. We let $\delta \stackrel{\text{def}}{=} \varepsilon/T$, where $\varepsilon = \varepsilon(n)$ will be set conveniently later in the proof. Now let

$$Q(x) \stackrel{\text{def}}{=} \sum_{i \in [T]} Q_i(x).$$

We claim that $Q \in \mathbb{F}_p[x_1, \ldots, x_n]$ is a polynomial that $(\varepsilon + \gamma)$-approximates MOD_q^n under the uniform distribution. Clearly,

$$\begin{aligned}
\Pr_{x \sim \{0,1\}^n}[\mathsf{MOD}_q^n(x) \neq Q(x)] &\leq \Pr\Big[\mathsf{MOD}_q^n(x) \neq \bigvee_{i \in [T(n)]} C_i(x)\Big] + \Pr\Big[\bigvee_{i \in [T(n)]} C_i(x) \neq Q(x)\Big] \\
&\leq \gamma + \Big(1 - \Pr\Big[\bigvee_{i \in [T(n)]} C_i(x) = Q(x)\Big]\Big).
\end{aligned}$$

For each $i \in [T]$, let $S_i \stackrel{\text{def}}{=} \{x \in \{0,1\}^n \mid Q_i(x) \neq C_i(x)\}$ be the set of bad inputs for Q_i, and set $S \stackrel{\text{def}}{=} \bigcup_{i\in[T]} S_i$. In order to complete the proof of our claim, we argue next that for every $y \notin S$, $Q(y) = \bigvee_{i\in[T(n)]} C_i(y)$.

First, if $\bigvee_{i\in[T(n)]} C_i(y) = 0$, then $Q_i(y) = 0$ for every $i \in [T]$, and we get $Q(y) = 0$. On the other hand, if $\bigvee_{i\in[T(n)]} C_i(y) = 1$, using the disjointness assumption for the family of circuits, it follows that there is exactly one circuit with $C_i(y) = 1$. Since $y \notin S$, we get that $Q_i(y) = 1$, while $Q_j(y) = 0$ for every $j \neq i$. Consequently, we have $Q(y) = 1$. (Observe that the extra assumption over the family of circuits is crucial for this case, since the original circuits produce Boolean values, while Q is an $\mathbb{F}_p$-polynomial.) Overall, it follows that $\Pr[\bigvee_{i\in[T(n)]} C_i(x) = Q(x)] \geq (2^n - |S|) \cdot 2^{-n} \geq 1 - T \cdot \delta = 1 - \varepsilon$, which establishes our initial claim.

Therefore, for every $\varepsilon(n) > 0$, there exists a polynomial $Q \in \mathbb{F}_p[x_1, \ldots, x_n]$ that $(\varepsilon+\gamma)$-approximates the MOD_q^n function over the uniform distribution, where

$$\mathsf{deg}(Q) \;\leq\; ((\ell \cdot \log n)^d \cdot \log(1/\delta)) \;\leq\; (\ell \cdot \log n)^d \cdot (\log T + \log(1/\varepsilon)). \tag{4.1}$$

On the other hand, we obtain from Proposition 4.3.3 that for every $\varepsilon(n) \in [2^{-n}, 1/10q]$, and every large enough n,

$$\zeta \cdot \sqrt{n \cdot \log(1/(\varepsilon+\gamma))} \;\leq\; \mathsf{deg}(Q). \tag{4.2}$$

Our result follows by combining Equations 4.1 and 4.2. Observe that we are free to set $\varepsilon(n)$ in order to maximize our lower bound on T, depending on the value of γ. If $0 < \gamma \leq 1/20q$, the first case of Lemma 4.3.6 follows if we let $\varepsilon = 1/20q$. On the other hand, when $\gamma = 0$, we get that

$$\log T(n) \;\geq\; \frac{\zeta \cdot \sqrt{n \cdot \log(1/\varepsilon)} - \log(1/\varepsilon) \cdot (\ell \cdot \log n)^d}{(\ell \cdot \log n)^d},$$

and the second case of Lemma 4.3.6 now follows by setting $\varepsilon = \exp(-\Theta(n/\log^{2d} n))$. □

We are now ready to prove an essentially optimal communication lower bound for $\mathsf{AC}^0_d[p]$-compression games for $\mathsf{Majority}$.

Theorem 4.3.7. *Let p be a prime number. There exists a constant $c \in \mathbb{N}$ such that, for any $d \in \mathbb{N}$, and every $n \in \mathbb{N}$ sufficiently large, the following holds.*

(*i*) *Any* $\mathsf{AC}^0_d[p]$*-compression game for* $\mathsf{Majority}_n$ (*with any number of rounds*) *has communication cost at least* $n/(\log n)^{2d+c}$.

(*ii*) *There exists a single-round* AC^0_d*-compression game for* $\mathsf{Majority}_n$ *with communication cost at most* $n/(\log n)^{d-c}$.

Proof. The lower bound follows immediately from Proposition 4.3.2, Proposition 4.3.5, and Lemma 4.3.6 (*ii*). The upper bound is given by Lemma 4.3.1. □

For randomized compression games, we are able to generalize the lower bound for single-round protocols obtained by Chattopadhyay and Santhanam [45] to protocols with any number of rounds.

Theorem 4.3.8. *Let* p *and* q *be distinct primes. There exists a constant* $c \in \mathbb{N}$ *such that, for any* $d \in \mathbb{N}$*, and* $n \in \mathbb{N}$ *sufficiently large, every randomized* $\mathsf{AC}^0_d[p]$*-compression game for* MOD^n_q *with any number of rounds and error at most* $1/3$ *has communication cost at least* $\sqrt{n}/(\log n)^{d+c}$.

Proof. If there exists a randomized compression protocol with these properties, we can boost its success probability to $1 - 1/20q$ on every input by repeating it a constant number of times, and applying a majority vote. Observe that the communication increases by a constant factor only, and that the majority vote can be computed efficiently, as it is over a constant number of bits. Since any randomized protocol with this success probability provides an average-case protocol that is correct on at least a $(1 - 1/20q)$-fraction of the inputs under the uniform distribution, the result follows from Proposition 4.3.5 and Lemma 4.3.6 (*i*). □

We stress that the results in Theorems 4.3.7 and 4.3.8 hold both for $\mathsf{Majority}$ and MOD_q, but we restricted each statement to a particular function for simplicity. In order to see this, first notice that the proof of Theorem 4.3.7 includes the argument for MOD_q. On the other hand, in order to extend Theorem 4.3.8 to $\mathsf{Majority}$, we can employ a reduction through Proposition 4.3.2. A subtle point is that for probabilistic protocols one has to make sure that the final error probability after the reduction is bounded. However, this can be

achieved during the proof by boosting the correctness probability of the initial protocol for $\mathsf{Majority}$ via repetition.

The proof of Theorem 4.3.7 can be generalized to an essentially optimal bound for $\mathsf{AC}^0_d[p](s(n))$-compression games computing MOD^n_q. The argument implies that this function has communication cost $n/(\log s)^{\Theta(d)}$. Observe that the original circuit size lower bounds obtained by Razborov [157] and Smolensky [177] follows from the analysis of communication protocols for $\mathsf{Majority}$ and MOD_q with constant communication cost. Interestingly, the polynomial method interpolates between essentially optimal communication lower bounds and circuit size lower bounds when applied with exponentially small error and constant error, respectively.

4.4 Multiparty interactive compression

The communication cost of k-party $\mathsf{AC}^0[p]$-compression games. We will prove in this section that $\mathsf{Majority}_n$ requires $\tilde{\Omega}(n^{1/2r})$ communication in the $(k+1)$-party r-round $\mathsf{AC}^0[p]$-compression game, for any $k = \mathsf{poly}(n)$. Put another way, although Alice is allowed to send roughly $n^{1/2r}$ bits to each individual Bob, even if n^{100} such parties are present, she will not be able to combine their answers in order to compute $\mathsf{Majority}_n$.

We start with the following upper bound, which can be seen as the corresponding analogue of Lemma 4.3.1.

Lemma 4.4.1. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be an arbitrary symmetric function, and p be any prime. For any $r \in \mathbb{N}$, f admits an $(\lceil n^{1/r} \rceil + 1)$-party r-round AC^0-compression game with cost $O(rn^{1/r}\log(n))$.*

Proof. We set up some notation first. Given n and r, let $T_{n,r}$ be the complete $\lceil n^{1/r} \rceil$-ary tree of depth r. We assume the leaves of $T_{n,r}$ to be ordered from left to right. Given an input x of length n, label the leaves of $T_{n,r}$ with bits of x in the natural way: the leftmost leaf is labelled with the first bit of x, the second to leftmost with the second bit, etc. Note that some leaves may remain unlabelled in this process.

Let V_d be the set of nodes at depth d in this tree, where $0 \le d \le r$. The protocol will proceed with Alice iteratively labelling nodes in the tree with numbers in $[n]$, each

node being labelled with the sum of all the leaves in the subtree rooted at the node. Any unlabelled leaf is assumed to have label 0. After round i, where $0 \leq i \leq r$, all nodes at depth $r-i$ or greater will be labelled. Once the root is labelled, Alice can compute $f(x)$ by herself, as $f(x)$ is purely a function of the label at the root (which is the weight of the input x), and any function of $O(\log n)$ bits can be computed in AC^0_2.

We assume inductively that after round i, all nodes at depth $r-i$ or greater have been labelled. The base case $i = 0$ clearly holds, as Alice can label the leaves herself. Assume that the inductive hypothesis holds after round i, where $0 \leq i < r$. We show it holds after round $i+1$. In round $i+1$, Alice arbitrarily associates a unique Bob with each node $v \in V_{r-i-1}$. This can be done as long as the number of parties is greater than $\lceil n^{1/r} \rceil$, as assumed. We denote the Bob associated with v by $\mathrm{Bob}(v)$. For each v, Alice sends to $\mathrm{Bob}(v)$ the sequence of labels of the children of v. Note that by the inductive assumption, the children of v have already been labelled. For each v, $\mathrm{Bob}(v)$ responds with the sum of all the integer labels sent by Alice to $\mathrm{Bob}(v)$ in the $(i+1)$-th round.

This is clearly a correct protocol. In any one round, Alice sends at most $\lceil n^{1/r} \rceil \cdot \lceil \log(n+1) \rceil$ bits to any Bob, as the number of children of any node in the tree is at most $\lceil n^{1/r} \rceil$, and each labelled node has a label in $[n]$. Thus, the cost of the protocol is $O(rn^{1/r} \log n)$, as claimed. □

Our lower bound is also based on algebraic arguments, but it employs a slightly different approach to that in the previous section. In particular, it does not rely on Proposition 4.3.5. We will need the following result.

Proposition 4.4.2 ([157])**.** *Let p be a fixed prime, and $P(x_1, \ldots, x_n) \in \mathbb{F}_p[x_1, \ldots, x_n]$ be a degree-ℓ polynomial. Then,*

$$\Pr_{x \sim \{0,1\}^n}[\mathsf{Majority}_n(x) = P(x)] \leq 1/2 + O(\ell/\sqrt{n}).$$

The next lemma allows us to construct low-degree probabilistic polynomials from multiparty compression games.

Lemma 4.4.3. *Let $\Phi_n^{[k]}$ be a randomized $(k+1)$-party r-round $\mathsf{AC}^0_d[p](\mathsf{poly}(n))$-compression protocol with signature $(n, s_1, t_1, \ldots, s_r, t_r)$ computing a Boolean function $h\colon \{0,1\}^n \to$*

$\{0,1\}$ *with error* γ*, where* $s_i \leq n$ *for each* $i \in [r]$*, and* $r \in \mathbb{N}$*. Then, for every* $\delta > 0$*,* h *admits a* $(\gamma+\delta)$*-error probabilistic polynomial over* $\mathbb{F}_p$ *with degree* $O\big((\sum_{i\in[r]} s_i)^r \cdot ((\log n)^{d+r} \cdot (\log 1/\delta))^{r+1}\big)$.

Proof. We start with a proof of the lemma for $r = 1$ and deterministic protocols that are always correct, then observe that the same proof can be generalized to randomized r-round protocols.

Suppose $\Phi_n^{[k]}$ is a $(k+1)$-party 1-round $\mathsf{AC}^0_d[p](\mathsf{poly}(n))$-compression protocol with signature (n, s_1, t_1) for a Boolean function h on inputs x of n bits. For each $i \in [k]$, let $a_1^i \dots a_{n_i}^i$ be the message bits sent by Alice to Bob_i in the first round, and let $b_1^i \dots b_{m_i}^i$ be Bob_i's response. Let a be the bit output by Alice at the conclusion of the protocol. By the definition of signature, we have that for each $i \in [k]$, $n_i \leq s_1$ and $m_i \leq t_1$. We also have that $a = 1$ if and only if $h(x) = 1$.

Each of the message bits sent by Alice in the first round is a function of x, and since Alice is $\mathsf{AC}^0_d[p](\mathsf{poly}(n))$-bounded, we can use Proposition 4.3.4 to obtain ε-error probabilistic polynomials $P_j^i \in \mathbb{F}_p[x_1, \dots, x_n]$, where $i \in [k], j \in [n_i]$, for each of these message bits. The degree of each polynomial is at most $d_1 = O((\log n)^{d-1} \cdot \log 1/\varepsilon)$, where $\varepsilon > 0$ is a parameter to be determined later. Since each message bit of each Bob_i is a function of the message bits sent by Alice to Bob_i, we can express each bit b_j^i of Bob_i as an *exact* polynomial Q_j^i in the message bits of Alice. Notice that each such polynomial has degree at most s_1. Now, again by Proposition 4.3.4, there is an ε-error probabilistic polynomial P of degree at most $d_2 = O((\log n)^{d-1} \cdot \log 1/\varepsilon)$ for a as a function of x, the message bits sent by Alice in the first round, and the message bits sent by each Bob in the first round.

If we set $\varepsilon = \delta/(s_1 \cdot k + 1)$, by using the union bound, we have that

$$P' \stackrel{\text{def}}{=} P(x, P_1^1(x), \dots, P_{n_k}^k(x), Q_1^1(P_1^1(x), \dots, P_{n_1}^1(x)), \dots, Q_{m_k}^k(P_1^k(x), \dots, P_{n_k}^k(x)))$$

is a δ-error probabilistic polynomial for h as a function of x. The degree of P' is at most $d_1 \cdot s_1 \cdot d_2 = O(s_1 \cdot ((\log n)^d \cdot \log 1/\delta)^2)$, where we have used that $\log 1/\varepsilon = O(\log n \cdot \log 1/\delta)$ due to the upper bound on s_1 and $k \leq \mathsf{poly}(n)$. This completes the proof for (deterministic) single-round protocols.

The proof for deterministic protocols with $r \geq 2$ rounds is by induction on the number

of rounds. Let $\Phi_n^{[k]}$ be a $(k+1)$-party r-round $\mathsf{AC}^0_d[p](\mathsf{poly}(n))$-compression protocol with signature $(n, s_1, t_1, \ldots, s_r, t_r)$ for a Boolean function h. Observe that during the last round of the protocol, each Bob_ℓ receives a message containing at most $s \stackrel{\text{def}}{=} \sum_{i\in[r]} s_i$ bits (recall that Bob_ℓ has access to the messages he received from Alice in previous rounds, and to no other message). We can view each bit a_j^ℓ of each such message as a Boolean function computed by a $(k+1)$-party $(r-1)$-round protocol, where $\ell \in [k]$, and $j \le s$. It follows from the induction hypothesis that there is a probabilistic polynomial $P_j^\ell \in \mathbb{F}_p[z_1, \ldots, z_{s'}]$ for an appropriate $s' \le s$ of degree at most

$$d_1 \;\le\; O\big(s^{r-1} \cdot ((\log n)^{d+(r-1)} \cdot (\log 1/\varepsilon))^r\big)$$

that ε-approximates a_j^ℓ, where $\varepsilon > 0$ will be set conveniently later in the proof.[1] Further, during the last round of the protocol, each bit b_j^ℓ sent by Bob_ℓ can be computed exactly by a (deterministic) polynomial Q_j^ℓ of degree at most s. Finally, the last bit output by Alice during the execution of $\Phi_n^{[k]}$ is computed by an $\mathsf{AC}^0_d[p]$ circuit over polynomially many input bits. According to Proposition 4.3.4, it can be ε-approximated by a probabilistic polynomial $P \in \mathbb{F}_p[y_1, \ldots, y_{\mathsf{poly}(n)}]$ of degree $d_2 \le O((\log n)^{d-1} \cdot \log 1/\varepsilon)$.

We now compose these polynomials appropriately, similarly to the base case, in order to obtain a probabilistic polynomial $P' \in \mathbb{F}_p[x_1, \ldots, x_n]$ that approximates the original Boolean function h compressed by $\Phi_n^{[k]}$. If we set $\varepsilon \stackrel{\text{def}}{=} \delta/(sk+1) = \delta/\mathsf{poly}(n)$, we get via an union bound that P' is a probabilistic polynomial that δ-approximates h. Finally, the degree of P' is upper bounded by

$$\begin{aligned} d_1 \cdot s \cdot d_2 &\le O\big(s^{r-1} \cdot ((\log n)^{d+(r-1)} \cdot (\log 1/\varepsilon))^r \cdot s \cdot (\log n)^{d-1} \cdot \log 1/\varepsilon\big) \\ &\le O\big(s^r \cdot ((\log n)^{d+r} \cdot (\log 1/\delta))^r \cdot (\log n)^d \cdot \log 1/\delta\big) \\ &\le O\big((\Sigma_{i\in[r]} s_i)^r \cdot ((\log n)^{d+r} \cdot (\log 1/\delta))^{r+1}\big), \end{aligned}$$

which completes the induction step.

It remains to handle the case of randomized protocols. Observe that for every fixed setting of the randomness of Alice, we obtain a multiparty compression protocol computing

[1] Our abuse of the asymptotic notation in this inductive proof is harmless, as we are proving the result for a fixed number of rounds only.

some Boolean function h_r. We can apply the procedure described above to get a probabilistic polynomial $P_r \in \mathbb{F}_p[x_1, \ldots, x_n]$ that agrees with h_r on every input $x \in \{0,1\}^n$ except with probability δ. Since over the choice of r we know that $h(x) = h_r(x)$ except with probability γ, we can obtain from the family of distributions P_r a single distribution over polynomials of the same degree that agrees with h on every input x except with probability $\gamma + \delta$, which completes the proof. □

We now have all ingredients to prove the main result of this section.

Theorem 4.4.4. *Let $p \in \mathbb{N}$ be a fixed prime. For every $k, r, d \in \mathbb{N}$, the following holds.*

(*i*) *There exists a deterministic $n^{1/r}$-party r-round $\mathsf{AC}^0[p]$-compression game for $\mathsf{Majority}_n$ with cost $O(n^{1/r} \cdot \log n)$.*

(*ii*) *Every randomized n^k-party r-round $\mathsf{AC}^0_d[p]$-compression game for $\mathsf{Majority}_n$ has cost $\Omega\big(n^{1/2r}/(\log n)^{2(d+r)}\big)$.*

Proof. The upper bound follows from Lemma 4.4.1. For the lower bound, assume $\Pi_n^{[k]}$ has signature $(n, s_1, t_1, \ldots, s_r, t_r)$ and satisfies the assumption of the theorem. Since $\Pi_n^{[k]}$ is a randomized protocol, we can reduce its error probability to $1/20$ by running it in parallel and computing a majority vote during the last round. Observe that the depth of the circuits used by Alice increases by at most 1 if this computation is performed by an appropriate DNF or CNF. Setting $\delta = 1/20$ in Lemma 4.4.3 and fixing the randomness, we can obtain an average-case (deterministic) polynomial for $\mathsf{Majority}_n$ of the stated degree and error $1/10$ with respect to the uniform distribution. Now applying Proposition 4.4.2 and using $1/\delta = O(1)$, we get that

$$(s_1 + s_2 + \ldots + s_r)^r \cdot (\log n)^{(d+r)(r+1)} \;\geq\; \Omega(\sqrt{n}),$$

which completes the proof of the lower bound, since $\mathsf{cost}(\Pi_n^{[k]}) = \sum_{i \in [r]} s_i$ and $r \geq 1$. □

As opposed to the statement of Theorem 4.3.7, we have not tried to optimize the logarithmic factors here, since there is still a polynomial gap in the bounds as a function of r.[2]

[2]For instance, in the proof of Lemma 4.4.1, it is possible to break the information passed to each Bob

Corollary 4.4.5. *For any* $r, \ell, d \in \mathbb{N}$*, the randomized* n^ℓ*-party* r*-round* $\mathsf{AC}^0_d[p]$*-compression cost of* $\mathsf{Majority}_n$ *is* $n^{\Theta(1/r)}$*.*

In addition, observe that Theorem 4.4.4 implies a round separation result for *multiparty* $\mathsf{AC}^0[p]$-compression games. In particular, we get the following consequence for single-round $\mathsf{AC}^0[p]$ protocols versus protocols with more rounds.

Corollary 4.4.6. *For every* $\varepsilon > 0$ *and* $\ell \in \mathbb{N}$*, there exists* $r \in \mathbb{N}$ *with* $r = O(1/\varepsilon)$ *for which the following holds, whenever* n *is sufficiently large. There exists an explicit function* $f_n \colon \{0,1\}^n \to \{0,1\}$ *such that*: f_n *admits no randomized* n^ℓ*-party single-round* $\mathsf{AC}^0[p]$*-compression games with cost* $n^{1/2-\varepsilon}$*, but it admits deterministic* n^ε*-party* r*-round* $\mathsf{AC}^0[p]$*-compression games of cost* n^ε*.*

Randomized versus deterministic games. Note that for two-party games we were able to obtain almost linear lower bounds for *deterministic* protocols (Theorem 4.3.7), while for probabilistic and average-case protocols we encountered a barrier at $c(n) \approx \sqrt{n}$ (Theorems 4.3.8 and 4.4.4). We are not aware of explicit lower bounds of the form $n^{1/2+\varepsilon}$ for a fixed $\varepsilon > 0$ for randomized two-party $\mathsf{AC}^0[p]$ games. It is natural to wonder if we can improve Theorem 4.4.4 in the case of *deterministic* k-party games.

We prove next that this is unlikely without the introduction of new ideas to handle probabilistic protocols. More precisely, we observe that k-party protocols can be derandomized without increasing communication cost. The proof relies on the definition of cost for such protocols as the length of the longest message sent by Alice to any particular Bob, and on the fact that we are dealing with non-uniform protocols/circuits. The argument is based on parallel repetition and composition of k-party protocols with an approximate majority function. We provide the details next.

We say that a Boolean function $h_n \colon \{0,1\}^n \to \{0,1\}$ is an (ℓ_1, ℓ_2)-approximate majority if $h_n(x) = 0$ on every x with $|x|_1 \leq \ell_1$, and $h_n(x) = 1$ on every x with $|x|_1 \geq \ell_2$.

Proposition 4.4.7 ([6])**.** *There exists a family* $h = \{h_n\}_{n\in\mathbb{N}}$ *of Boolean functions in* $\mathsf{AC}^0_3(\mathsf{poly}(n))$ *for which every* h_n *is an* $(0.49n, 0.51n)$*-approximate majority.*

---

into multiple blocks as done in the proof of Lemma 4.3.1, and save an extra $(\log n)^{\Theta(d)}$ factor during each round by allowing Alice to make partial progress towards the computation of $\mathsf{Majority}$.

Theorem 4.4.8. *Let $\mathcal{C}$ be a circuit class, $d \geq 1$, and $f = \{f_n\}_{n\in\mathbb{N}}$ be a family of Boolean functions, where $f_n \colon \{0,1\}^n \to \{0,1\}$. Suppose f admits a k-party probabilistic $\mathcal{C}_d(\mathsf{poly}(n))$-compression game with cost $c(n)$ and error $\gamma(n) \leq 1/3$, where $k = O(\mathsf{poly}(n))$. Then f admits a k'-party deterministic $\mathcal{C}_{d+O(1)}(\mathsf{poly}(n))$-compression game with the same cost $c(n)$ and $k' = O(\mathsf{poly}(n))$.*

Proof. By assumption, f has a k-party probabilistic $\mathcal{C}_d(\mathsf{poly}(n))$-compression protocol Π with cost $c(n)$ and error $\gamma(n) \leq 1/3$, where $k = O(\mathsf{poly}(n))$. We define a new probabilistic protocol for f with the same cost but with $k' \stackrel{\text{def}}{=} \ell n \cdot k$ parties and with error $\gamma'(n) < 2^{-n}$, where $\ell > 0$ is a constant which we determine later. We then use Adleman's trick to fix the random bits used by Alice, thus making the protocol deterministic.

The new probabilistic protocol Π' for f simply simulates ℓn copies of the protocol Π in parallel. Namely, we interpret the Bobs to be partitioned into ℓn sets, each of size k, and Alice independently executes the protocol in parallel for each set of Bobs. Note that by our definition of cost, the cost for each round of Π' is the same as the cost for each round of Π. In the final step of the protocol, Π' applies the Approximate Majority function $h_{\ell n}$ to the answers of Π for the ℓn parallel executions. Using Proposition 4.4.7, Alice can be implemented to work in $\mathcal{C}_{d+O(1)}(\mathsf{poly}(n))$. It follows by a standard application of Proposition 2.0.1 that if we set ℓ to be a large enough constant, the error probability of the new protocol Π' is strictly less than 2^{-n}.

Now, there must exist some setting of the random bits of Alice that yields the correct answer for *every* $x \in \{0,1\}^n$, simply by using the union bound. By fixing the random bits of Alice accordingly, we derive a *deterministic* protocol with cost $c(n)$, which completes the proof. □

4.5 The connection with circuits augmented with oracle gates

In this section we observe that lower bounds on interactive compressibility are closely connected to lower bounds against oracle circuits with arbitrary oracles. We first show such a connection for 2-party compression games, and then for multiparty compression games.

In order to formalize these connections, we need to define classes of oracle circuits

corresponding to classes of Boolean circuits. Such a definition is especially non-obvious for bounded-depth circuit classes – should we consider oracle gates when counting the depth or not? We use a very generous notion of oracle circuits. We say that an oracle circuit C belongs to the oracle analogue of a Boolean circuit class $\mathcal{C}$ if every maximal subcircuit of C without oracle gates belongs to $\mathcal{C}$. Put another way, every subcircuit induced by a connected subgraph of the acyclic graph encoding C that does not contain an oracle gate is a circuit from $\mathcal{C}$. The generosity of this notion only makes the lower bounds we derive from the connections below stronger.

For the sake of convenience, we abuse notation and occasionally use $\mathcal{C}$ to refer both to a Boolean circuit class and its oracle analogue.

Proposition 4.5.1. *Let $\mathcal{C}$ be a circuit class. Let C be an oracle circuit over n variables from $\mathcal{C}(\mathsf{poly}(n))$ with oracle gates $f_i\colon \{0,1\}^{s_i} \to \{0,1\}^{t_i}$, where $i \in [r]$, for some $r = r(n)$. In addition, let $s = s_1 + \ldots + s_r$ be the total fan-in of these oracle gates, and $h\colon \{0,1\}^n \to \{0,1\}$ be the Boolean function computed by C. Then h admits a $\mathcal{C}(\mathsf{poly}(n))$-compression game with communication cost $c(n) \leq s+1$ consisting of at most $r+1$ rounds.*

Proof. We describe a protocol for the compression game for h in which Alice sends at most $s+1$ bits to Bob, and where each of Alice's messages is computable by a small circuit from $\mathcal{C}$.

First Alice topologically sorts the circuit C with respect to oracle gates, namely she constructs a graph G whose nodes are the oracle gates of the circuit, and there is an edge from a node u to a node v if and only if there is a path from the oracle gate represented by u to the oracle gate represented by v in the digraph C. The graph G is a DAG, and hence its vertices can be topologically sorted. Let $g_1, g_2 \ldots g_r$ be the topological ordering of the oracle gates. Alice proceeds inductively as follows. In round i, where $i \in [r]$, she computes all inputs to the gate g_i using her input x and previous messages sent by Bob. She then sends the values of these input bits to Bob, who in turn computes the value of the gate g_i applied to these bits, and sends her the answer. Note that g_1 has no predecessors which are oracle gates, and therefore Alice can compute all the inputs to g_1 herself using circuits from $\mathcal{C}$ (which are sub-circuits of C) applied to the input x. Gate g_i only has gates $g_1 \ldots g_{i-1}$ as predecessors, and by the definition of the protocol, Alice has already received the values of

these gates from Bob in previous rounds, hence she can calculate values of inputs to g_i from x and previous messages using circuits from $\mathcal{C}$. In round $r+1$, Alice computes the value of the circuit C on x and sends it to Bob, thus completing the protocol.

The total number of bits sent by Alice to Bob is the total fan-in of the oracle gates plus one, i.e., $s+1$, and there are $r+1$ rounds in the protocol. □

Note that Proposition 4.5.1 only gives useful information when the total fan-in of oracle gates is sub-linear. We'd like to also show lower bounds on oracle gates where the total fan-in is not bounded in this way. This is where multiparty compression games, and the modified notion of protocol cost for such games, come in useful.

We need some more terminology for oracle circuits. An oracle circuit C has r layers if the oracle gates can be partitioned into r sets such that no two gates within any set are connected by a path in C. Equivalently, there are at most r oracle gates on any path from an input of C to the output.

Proposition 4.5.2. *Let D be an oracle circuit over n variables from $\mathcal{C}(\mathsf{poly}(n))$ augmented with r layers of oracle gates, where for each $i \in [r]$, s_i is the maximum fan-in of a gate in the i-th layer, and where there are at most k gates in each layer. Let $s = \sum_{i\in[r]} s_i$. In addition, let $h\colon \{0,1\}^n \to \{0,1\}$ be the Boolean function computed by D. Then h admits a $(k+1)$-party $\mathcal{C}(\mathsf{poly}(n))$-compression game with r rounds and communication cost $c(n) \leq s$.*

Proof. Alice orders the layers of oracle gates topologically, so that there are no paths from gates in layer i to gates in layer j for $i > j$. The protocol proceeds with Alice inductively computing all input bis to oracle gates in the i-th layer, where $i \in [r]$, and then delegating the computations of gates in the i-th layer to the Bobs, a different Bob for each oracle gate. Since there are at most k gates in each such layer, she can successfully assign a different Bob to each oracle gate in any specific layer. Alice can compute all inputs to an oracle gate in the first layer by herself, as all of these can be computed by circuits in $\mathcal{C}(\mathsf{poly}(n))$. In the i-th round, where $i \in [r]$, Alice chooses a different Bob for each oracle gate in layer i, and sends to the corresponding Bob the values of the inputs to the corresponding gate. She can compute these values using circuits in $\mathcal{C}$, as the output bits of all oracle gates in layer $i-1$ or below are already known to her by the definition of the protocol. The Bob

corresponding to a gate responds with the output values of that gate. After the r-th round, Alice computes the output value of the circuit C, and outputs it.

Notice that Alice sends at most s_i bits to any individual Bob in round i by our assumption on the fan-in of oracle gates in C. Thus the cost of the protocol is s. It is clear that the protocol operates in r rounds. □

Observe that Propositions 4.5.1 and 4.5.2, together with Theorems 4.3.7 and 4.4.4, imply strong limitations on the progress that $\mathsf{AC}^0[p]$ circuits can make towards the goal of computing the Majority function. In particular, a circuit of this form extended with arbitrary oracle gates can only compute $\mathsf{Majority}_n$ if it delegates essentially all the work to these extra gates. We can formalize this claim as follows.

Corollary 4.5.3. *Let $p \geq 2$ be prime, and $d \in \mathbb{N}$. There exists a constant $c \in \mathbb{N}$ such that, for every sufficiently large n, the following holds. If $\mathsf{Majority}_n$ is computed by $\mathsf{AC}^0_d[p]$ circuits of polynomial size with arbitrary oracle gates, then the total fan-in of the oracle gates is at least $n/(\log n)^{2d+c}$.*

Proof. This result follows immediately from Proposition 4.5.1 and Theorem 4.3.7. The fan-in lower bound is independent of the number of oracle gates, as Theorem 4.3.7 holds for protocols with any number of rounds. □

This result has an interesting consequence on the structure of $\mathsf{AC}^0[p]$ circuits computing $\mathsf{Majority}$. More precisely, Corollary 4.5.3 implies that in any layered circuit computing $\mathsf{Majority}_n$, at least $\lfloor n/(\log n)^{O(k)} \rfloor$ gates must be present at the k-th layer of the circuit (in order to see this, transform the circuit into an equivalent circuit with a single oracle gate at the top after the first k layers). On the other hand, the construction in Lemma 4.3.1 shows that this bound is not far from optimal. A similar consequence holds for polynomial size circuits computing the MOD_q function.

Using Proposition 4.5.2 and Theorem 4.4.4, we derive lower bounds on the maximum fan-in of oracle gates in oracle circuits with a bounded number of such layers computing $\mathsf{Majority}$. The *number* of oracle gates is now allowed to be polynomially large.

Corollary 4.5.4. *Let $p \geq 2$ be prime, and $r, d \in \mathbb{N}$. If $\mathsf{Majority}_n$ is computed by an $\mathsf{AC}^0_d[p]$ circuit of polynomial size with arbitrary oracle gates that contains at most r layers of such gates, then there is some oracle gate with fan-in at least $n^{1/2r}/\mathsf{polylog}(n)$.*

Proposition 4.5.2 suggests an approach to the NP vs. $\mathsf{NC}^1/\mathsf{poly}$ problem. The key observation is that for any r, every Boolean function in $\mathsf{NC}^1/\mathsf{poly}$ has oracle circuits of polynomial size with r layers, where the maximum fan-in of any oracle gate is $n^{O(1/r)}$.

Proposition 4.5.5. *Let $f = \{f_n\}_{n\in\mathbb{N}}$ be a family of Boolean functions in $\mathsf{NC}^1/\mathsf{poly}$, and $r \in \mathbb{N}$. Then f has AC^0 oracle circuits of polynomial size with r layers, where the maximum fan-in of any oracle gate is $n^{O(1/r)}$.*

Proof. Let $\{C_n\}_{n\in\mathbb{N}}$ be a sequence of circuits for f, where each C_n has size at most n^k and depth at most $c \log n$, for fixed constants k and c. We define oracle circuits D_n as follows. Divide C_n into r equally spaced layers of gates, with the distance between any two layers being at most $(c/r) \log n$. Replace each node at a layer boundary by an oracle gate whose inputs are its predecessors on the previous layer boundary. Note that any oracle gate has at most $n^{c/r}$ inputs, since the circuit has bounded fan-in. There are clearly a polynomially bounded number of oracle gates. Also, the circuit is an AC^0 circuit, since it consists purely of inputs and oracle gates. □

Applying Proposition 4.5.2 yields the following corollary.

Corollary 4.5.6. *Let r be any positive integer. Every function in $\mathsf{NC}^1/\mathsf{poly}$ admits $\mathsf{poly}(n)$-party $\mathsf{AC}^0(\mathsf{poly}(n))$-compression games with r rounds and cost $n^{O(1/r)}$.*

Thus a stronger lower bound than in Corollary 4.5.4 for an explicit function in NP would imply a separation of NP and $\mathsf{NC}^1/\mathsf{poly}$. We conjecture that Clique is such a function.

4.6 Interactive compression versus computation

The results of this chapter and in [45] show that two important techniques in circuit complexity, namely, random restrictions and approximation by low-degree polynomials, can be used to prove strong incompressibility lower bounds. It is natural to wonder if other

important lower bounds in complexity theory can be extended in a similar way. A related problem is whether compression can be easier than exact computation. Our next result sheds more light into these questions.

Let $\mathsf{IP}_n \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the Inner Product function. In other words, for $x, y \in \{0,1\}^n$, $\mathsf{IP}_n(x,y) \stackrel{\text{def}}{=} \sum_{i\in[n]} x_i \cdot y_i \pmod 2$. It is known that $\mathsf{IP}_n \notin \mathsf{THR} \circ \mathsf{MAJ}$, i.e., this function cannot be computed by polynomial size circuits consisting of a bottom layer of linear threshold functions with polynomial weights, connected to a top gate computed by an arbitrary linear threshold function ([69, 68]).[3]

We observe below that IP_n admits a $(\mathsf{MAJ} \circ \mathsf{MAJ})$-compression game with communication cost $O(\log n)$. In other words, there is a natural Boolean function that cannot be computed by certain circuits, but whose computation becomes feasible if Alice is allowed to interact with a more powerful party.

Proposition 4.6.1. *Let* $\mathsf{IP} = \{\mathsf{IP}_n\}_{n\in\mathbb{N}}$ *be the family of Inner Product functions. There exists a* $(\mathsf{MAJ} \circ \mathsf{MAJ})$*-compression game for* IP *with communication cost* $c(n) = O(\log n)$.

Proof. The protocol consists of $O(\log n)$ rounds, where in each round Alice sends a single bit, and Bob replies with a string $v \in \{0,1\}^n$. After the last round, Bob knows the sum $\sum_{i\in[n]} x_i \cdot y_i$, and therefore the transcript reveals the value $\mathsf{IP}_n(x,y)$. More details follow.

Alice's circuits are of the form $C(x,y,v)$. In the first layer of the circuit, C computes $z_i \stackrel{\text{def}}{=} x_i \wedge y_i$, for every $i \in [n]$. In the second layer, C outputs $\mathsf{sign}(\sum_{i\in[n]} z_i - v_i)$. Put another way, Alice uses the same circuit in every round, and we assume that the first bit sent by Alice during the first round is discarded. Bob does all the work, and simulates a binary search by sending to Alice an appropriate string v during each round. For instance, Bob sends $v = 0^{n/2}1^{n/2}$ during the first round, and the next bit computed by Alice reveals if $\sum_{i\in[n]} x_i \cdot y_i$ is at least $n/2$. After each round, Bob sends a string corresponding to the next step of the binary search, and so on. Clearly, after $O(\log n)$ rounds, Bob knows the value $\sum_{i\in[n]} x_i \cdot y_i$. Finally, observe that Alice communicates $O(\log n)$ bits, and that her circuits are of the form $\mathsf{MAJ} \circ \mathsf{MAJ}$. □

[3] Recall that a function $f \colon \{0,1\}^n \to \{0,1\}$ is a linear threshold function if there exist weights $w_1, \ldots, w_n \in \mathbb{Z}$ and a threshold $\theta \in \mathbb{Z}$ such that $f(x) = \mathsf{sign}(\sum_{i\in[n]} w_i \cdot x_i - \theta)$.

4.7 An improved round separation theorem for AC^0

Recall that Chattopadhyay and Santhanam [45] proved that there are Boolean functions on n variables that admit AC^0-bounded protocols with r rounds and cost $O(n^{1/r})$, but for which any correct AC^0-bounded $(r-1)$-round protocol has cost $\Omega(n^{2/r-o(1)})$. We use a different construction and refine their techniques, obtaining the following result.

Theorem 4.7.1. *Let $r \geq 2$ and $\varepsilon > 0$ be fixed parameters. There is an explicit family of functions $f = \{f_n\}_{n\in\mathbb{N}}$ with the following properties:*

(i) There exists an $\mathsf{AC}^0_2(n)$-bounded protocol Π_n for f_n with r rounds and cost $c(n) \leq n^{\varepsilon}$, for every $n \geq n_f$, where n_f is a fixed constant that depends on f.

(ii) Any $\mathsf{AC}^0(\mathsf{poly}(n))$-bounded protocol Π for f with $r-1$ rounds has cost $c(n) \geq n^{1-\varepsilon}$, for every $n \geq n_\Pi$, where n_Π is a fixed constant that depends on Π.

We will need some additional definitions and notation in order to establish this result. For any $n \in \mathbb{N}$, let $g_n \colon \{0,1\}^n \to \{0,1\}$ be the parity function on n variables, and $g = \{g_n\}_{n\in\mathbb{N}}$. Let m, ℓ, and r be positive integers. Set $n = n(m,\ell,r) \stackrel{\text{def}}{=} m + \ell \cdot r \cdot m$. We define a function $f_{m,\ell,r} \colon \{0,1\}^n \to \{0,1\}$ that will be used to prove round separation results for AC^0-compression games. For convenience, let $k \stackrel{\text{def}}{=} \log \ell$ and $v \stackrel{\text{def}}{=} m/\log \ell$. The definition of $f_{m,\ell,r}$ depends on g and a given function $h \colon \{0,1\}^k \to [\ell]$, which we assume to be some fixed one-to-one function.

Given any string $z \in \{0,1\}^n$, we write $z = (x, y^{(\cdot,1)}, \ldots, y^{(\cdot,r)})$, where $x \in \{0,1\}^m$, and $y^{(\cdot,j)} = (y^{(1,j)}, \ldots, y^{(\ell,j)})$, where $j \in [r]$, and $y^{(i,j)} \in \{0,1\}^m$, for every $i \in [\ell]$. In addition, for any string $w \in \{0,1\}^m$, we write $w = (w^{(1)}, \ldots, w^{(k)})$, where each $w^{(u)} \in \{0,1\}^v$, for $u \in [k]$. For convenience, instead of writing $y^{(i,j)(u)}$, we may also use $y^{(i,j,u)}$.

The function $f_{m,\ell,r}$ is defined by induction on r. It is simply a pointer jumping function, where h is applied to certain bits computed from the current string (initially x) using $k = \log \ell$ independent applications of g_v. After jumping from the initial x to a new string x', which will be one of the y's in $y^{(\cdot,1)}$, we recurse. After r steps, some string y from $y^{(\cdot,r)}$ will be reached. The output of $f_{m,\ell,r}$ is then set to be $g_m(y)$.

Formally, when $r = 1$, for any $z \in \{0,1\}^n$,

$$f_{m,\ell,1}(z) \stackrel{\text{def}}{=} g_m(y^{(i,1)}), \text{ where } i = h(g_v(x^{(1)}), \ldots, g_v(x^{(k)})).$$

Now let $r \geq 2$ be arbitrary. Then, for any $z \in \{0,1\}^n$,

$$f_{m,\ell,r}(z) \stackrel{\text{def}}{=} f_{m,\ell,r-1}(z'),$$
$$\text{where } z' = (x', y^{(\cdot,2)}, \ldots, y^{(\cdot,r)}),\ x' = y^{(i,1)}, \text{ and } i = h(g_v(x^{(1)}), \ldots, g_v(x^{(k)})).$$

This completes the definition of $f_{m,\ell,r}$.

Lemma 4.7.2 (Upper Bound)**.** *For any $m, \ell, r \geq 1$, the function $f_{m,\ell,r}$ admits an $\mathsf{AC}^0_2(m \cdot \ell)$-compression game with $r+1$ rounds and communication cost $c(n) = (r+1) \cdot m$.*

Proof. During each round j, Alice sends her current string $x' \in \{0,1\}^m$ to Bob, which replies with ℓ strings $v^{(i)} \in \{0,1\}^m$ satisfying the following property: $v^{(i)} = 1^m$ if the next round of the game is played on $y^{(i,j+1)}$, and $v^{(i)} = 0^m$ otherwise. Observe that the next message that Alice has to send is simply the m-bit string given by

$$\bigvee_{i \in [\ell]} \left(v^{(i)} \wedge y^{(i,j+1)} \right).$$

The cost and round complexity of this protocol is clear. □

We now proceed with the proof that in any AC^0-bounded protocol for $f_{m,\ell,r}$ with r rounds, Alice has to communicate roughly $\ell \cdot m$ bits, for an appropriate choice of ℓ that we would like to make as large as possible. The argument is based on random restrictions, which allow us to simplify the AC^0 circuits used by Alice considerably, while still maintaining the resulting function sufficiently hard for compression games. At a high level, we apply a round elimination technique, combined with a strong lower bound for $f_{m,\ell,1}$. More details follow.

From now on we will also view $f_{n,\ell,r}$ as a function $f_{m,\ell,r} \colon \{0,1\}^{[n]} \to \{0,1\}$, where each input z for $f_{m,\ell,r}$ can also be interpreted as a function $z \colon [n] \to \{0,1\}$. This will give us more flexibility when manipulating restrictions. A *restriction* $\rho \in \{0,1,*\}^{[n]}$ is simply a function $\rho \colon [n] \to \{0,1,*\}$. Given a restriction ρ and a function $f \colon \{0,1\}^{[n]} \to \{0,1\}$, we let $f^\rho \colon \{0,1\}^{\rho^{-1}(*)} \to \{0,1\}$ be the following function. For every $z^- \in \{0,1\}^{\rho^{-1}(*)}$,

$$f^\rho(z^-) \stackrel{\text{def}}{=} f(z),$$

where $z \in \{0,1\}^{[n]}$ is the function with $z|_{\rho^{-1}(\{*\})} = z^-$ and $z|_{\rho^{-1}(\{0,1\})} = \rho|_{\rho^{-1}(\{0,1\})}$.

Let $N \stackrel{\text{def}}{=} [n]$. Recall that we write $z \in \{0,1\}^n$ as $z = (x, y^{(1,1)}, \ldots, y^{(\ell,r)})$. Similarly, we let $S^{(i,j,u)} \subseteq N$ index the variables corresponding to $y^{(i,j,u)}$, for $i \in [\ell]$, $j \in [r]$ and $u \in [k]$. We define $S^{(i,j)} \stackrel{\text{def}}{=} \bigcup_u S^{(i,j,u)}$. Further, we use $M \subseteq N$ to index the variables corresponding to x, and $M^{(1)}, \ldots, M^{(k)}$ for the corresponding variables $x^{(1)}, \ldots, x^{(k)}$. Let Γ_N be the set of all restrictions with domain N, i.e., $\Gamma_N \stackrel{\text{def}}{=} \{0,1,*\}^N$. Given $\rho_1, \rho_2 \in \Gamma_N$, we say that ρ_2 extends ρ_1 if $\rho_2^{-1}(*) \subseteq \rho_1^{-1}(*)$ and $\rho_2|_{\rho_1^{-1}(\{0,1\})} = \rho_1|_{\rho_1^{-1}(\{0,1\})}$.

Our round separation theorem will be derived from lower bounds on a class of functions $\phi_{s,d,\ell} \colon \mathbb{N} \times \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$, defined as follows:

$$\phi_{s,d,\ell}\big(m, r, \delta\big) \stackrel{\text{def}}{=} \min_{\sigma \in \Gamma_{N,\delta}} \; \min_{\Pi \in \mathsf{Prot}^{\sigma}_{s,d,r}} \mathsf{cost}(\Pi),$$

where:[4]

(*i*) $\Gamma_{N,\delta} \subseteq \Gamma_N$ is the set of all restrictions σ for which the following holds: there exists sets $D_j \subseteq [\ell]$ with $j \in [r]$ such that $|D_j| \leq \delta \cdot \ell$, and $\sigma^{-1}(\{0,1\}) = \bigcup_{j \in [r]} \Big(\bigcup_{i \in D_j} S^{(i,j)} \Big)$,

(*ii*) $\mathsf{Prot}^{\sigma}_{s,d,r}$ is the set of all $\mathsf{AC}^0_d(s)$-bounded r-round protocols Π solving the compression game of $f^{\sigma}_{m,\ell,r}$.

The parameters m, r, and δ will vary during our inductive proof, while s, d, and ℓ remain fixed (observe that this is reflected in our notation for ϕ). The proof of Theorem 4.7.1 relies on the following lemmas, whose proof we present later in this section.

Lemma 4.7.3 (Lower Bound: Base case)**.** *Let $s = n^{c_1}$, $d \in \mathbb{N}$, $\ell = m^{c_2}$, $\delta \in (0, 1/10)$, and $r = 1$, where c_1 and c_2 are fixed positive integers. Then, for every fixed $\beta \in (0, 1/10)$ and m sufficiently large,*

$$\phi_{s,d,\ell}\big(m, 1, \delta\big) \geq \ell \cdot m^{1-\beta}.$$

Lemma 4.7.4 (Lower Bound: Induction step)**.** *Let $s = n^{c_1}$, $d \in \mathbb{N}$, $\ell = m^{c_2}$, $\delta \in (0, 1/10)$, and $r \geq 2$, where c_1 and c_2 are fixed positive integers. Then, for every fixed $\beta \in (0, 1/10)$ and m sufficiently large,*

$$\phi_{s,d,\ell}\big(m, r, \delta\big) \geq \min\Big\{ \ell \cdot m^{1-\beta} \;,\; \phi_{s,d,\ell}\big(m^{1-\beta}, r-1, \delta+\beta\big) \Big\}.$$

[4]For the sake of this proof, we consider circuits of size at most s (exactly), instead of $O(s)$.

These lemmas imply the following result.

Proposition 4.7.5. *For every fixed $r \geq 1$, $c \in \mathbb{N}$, and $\zeta > 0$, for m sufficiently large, we have*

$$\phi_{\mathsf{poly}(n),O(1),m^c}\big(m, r, 1/(100r)\big) \geq \ell \cdot m^{1-\zeta}.$$

Proof. The result follows easily from Lemmas 4.7.3 and 4.7.4 using that r is constant and that we can take β and δ sufficiently small. □

Finally, it is not hard to derive the main lower bound of this section from these results.

Proof of Theorem 4.7.1. Given any $r \geq 2$ and $\varepsilon > 0$, it is enough to consider an appropriate family of functions $f_{m,\ell,r-1}$, where $c = c(\varepsilon)$ is sufficiently large, and set $\ell = m^c$. The result then follows from Lemma 4.7.2 and Proposition 4.7.5. □

We proceed now with the proof of the lemmas. We will need the notion of a random restriction. Let $p \in [0,1]$ be a real number. We let Γ_N^p denote the distribution over restrictions $\rho \in \Gamma_N$ generated by independently fixing each $\rho(i)$ (where $i \in N$) as follows:

$$\Pr[\rho(i) = *] = p, \quad \Pr[\rho(i) = 1] = (1-p)/2, \quad \Pr[\rho(i) = 0] = (1-p)/2.$$

Given a Boolean function $f_n \colon \{0,1\}^n \to \{0,1\}$ over n variables, we let $\mathsf{DT}_{\mathsf{depth}}(f)$ be the smallest decision tree depth among all decision trees computing f_n. The next statement is independent of the number of inputs of f.

Lemma 4.7.6 (Switching Lemma [95])**.** *Let f be a Boolean function that can be written as a conjunction or disjunction of any number of depth-t decision trees. Then, for every $p \in [0,1]$ and $r \in \mathbb{N}$,*

$$\Pr_{\rho \sim \Gamma^p}[\mathsf{DT}_{\mathsf{depth}}(f^\rho) > r] \leq (5pt)^r.$$

The next result is a standard consequence of Lemma 4.7.6 (cf. Gopalan and Servedio [88]).

Proposition 4.7.7. *Let f be a Boolean function computed by an AC^0 circuit of size M and depth d. For every $t \in \mathbb{N}$, if $p \leq 1/(10t)^d$ then*

$$\Pr_{\rho \sim \Gamma^p}[\mathsf{DT}_{\mathsf{depth}}(f^\rho) > t] \leq M \cdot 2^{-t}.$$

Given a function $C\colon \{0,1\}^{[n]} \to \{0,1\}$, we let $\mathsf{live}(C) \subseteq [n]$ denote the set of input variables of C with influence greater than zero. It will be more convenient for us to rely on the following straightforward consequence of Lemma 4.7.6 and Proposition 4.7.7.

Lemma 4.7.8. *Let $C_1, \ldots, C_{s_1}\colon \{0,1\}^{n_1} \to \{0,1\}$ be functions computed by depth-d AC^0 circuits of size at most $n_1^{c_1}$, where $d, c_1 \in \mathbb{N}$ and $s_1 = m^{1-\gamma} \cdot \ell$, and these parameters satisfy $m, \ell \in \mathbb{N}$, $\gamma \in (0, 1/5)$, $\ell = m^{c_2}$, where $c_2 \in \mathbb{N}$, and $n_1 = \Theta(m \cdot \ell)$. Then, for $p = m^{-\gamma/2}$, there exists a constant c_3 such that, as $m \to \infty$,*

$$\Pr_{\rho \sim \Gamma^p_{[n_1]}} \left[\left| \bigcup_{i \in [s_1]} \mathsf{live}\left(C_i^\rho\right) \right| \leq c_3 \cdot (m^{1-\gamma} \cdot \ell) \right] \to 1.$$

Proof. Let $p = p_1 \cdot p_2$, where $p_1 = p_2 = m^{-\gamma/4}$. Observe that sampling a restriction $\rho \sim \Gamma^p_{[n_1]}$ is equivalent to first sampling some $\rho_1 \sim \Gamma^{p_1}_{[n_1]}$, followed by a restriction $\rho_2 \sim \Gamma^{p_2}_W$, where $W \stackrel{\text{def}}{=} [n_1] \setminus \rho_1^{-1}(\{0,1\})$, and finally setting $\rho = \rho_2 \circ \rho_1$, where the composition operation is defined in the natural way. Let $c = c_1 + 10$, and $t = c \cdot \log n_1$. Furthermore, we let $r = \lceil 8(1 + c_2)/\gamma \rceil$, and $c_3 = 2^r$. Then,

$$\begin{aligned} \Pr_{\rho \sim \Gamma^p_{[n_1]}} \left[\left| \bigcup_{i \in [s_1]} \mathsf{live}\left(C_i^\rho\right) \right| > c_3 \cdot (m^{1-\gamma} \cdot \ell) \right] &\leq \Pr_{\rho \stackrel{\text{def}}{=} \rho_2 \circ \rho_1} \left[\exists i \in [s_1] \text{ s.t. } |\mathsf{live}(C_i^\rho)| > 2^r \right] \\ &\leq \Pr_{\rho_1, \rho_2} \left[\exists i \in [s_1] \text{ s.t. } \mathsf{DT}_{\mathsf{depth}}(C_i^\rho) > r \right] \end{aligned}$$

In order to conclude the proof, it is enough to show that for every $j \in [s_1]$ and sufficiently large m, $\Pr_{\rho_1,\rho_2}[\mathsf{DT}_{\mathsf{depth}}(C_j^\rho) > r] \leq (1/n_1)^2$. However, by our choice of parameters (and with room to spare), this follows from an application of Proposition 4.7.7 with ρ_1 and t, followed by an application of Lemma 4.7.6 with ρ_2 and r (notice that these statements are true with respect to any input size). □

We are now ready to prove Lemmas 4.7.3 and 4.7.4.

Proof of Lemma 4.7.3. Let $\sigma\colon [n] \to \{0,1,*\}$ be a restriction in $\Gamma_{N,\delta}$, where $n = m + \ell \cdot m$ and $N = [n]$, as usual. Let $N_1 \stackrel{\text{def}}{=} N \setminus \sigma^{-1}(\{0,1\})$, and set $n_1 \stackrel{\text{def}}{=} |N_1|$. Observe that $n_1 \geq (1-\delta) \cdot \ell \cdot m = \Theta(m \cdot \ell)$. In addition, let $\Pi = (C^{(1)}, g^{(1)}, E)$ be a single-round protocol for $f^\sigma_{m,\ell,1}$, where $C^{(1)} = (C_1, \ldots, C_{s_1})$, and these are AC^0 circuits of depth d and size $s = n^{c_1} \leq n_1^{2c_1}$ (for large enough m) that compute the message in $\{0,1\}^{s_1}$ that Alice

sends to Bob. By definition, for each $i \in [s_1]$, $C_i \colon \{0,1\}^{n_1} \to \{0,1\}$. We prove that if $s_1 < \ell \cdot m^{1-\beta}$, then there exists an input $z \in \{0,1\}^{n_1}$ for which $\Pi(z) \neq f^{\sigma}_{m,\ell,1}(z)$.

Let $D_1 \subseteq [\ell]$ be the set identifying the variables y fixed by σ (according to our definition of $\Gamma_{N,\delta}$). For any $z \in \{0,1\}^{N_1}$, we write $z = (x, y^{(i_1,1)}, \ldots, y^{(i_k,1)})$, where $[\ell] \setminus D_1 = \{i_1, \ldots, i_k\}$, $k \geq (1-\delta) \cdot \ell$, and $x \in \{0,1\}^m$. Recall that we use sets $S^{(i_1,1)}, \ldots, S^{(i_k,1)}$ and M to address the elements of $[N_1]$ corresponding to these input positions.

Now consider a random restriction $\rho \sim \Gamma^p_{N_1}$, where $p = m^{-\beta/2}$. Applying Lemma 4.7.8 with $\gamma = \beta$ and Proposition 2.0.1, it follows that, for every large enough m, with high probability:

(*i*) $C^{(1),\rho}$ depends on at most $O(m^{1-\beta} \cdot \ell)$ variables.

(*ii*) For every $j \in [\log \ell]$, it is the case that $\rho^{-1}(*) \cap M^{(j)} \neq \emptyset$.

(*iii*) $|\rho^{-1}(*) \cap (S^{(i_1,1)} \cup \ldots \cup S^{(i_k,1)})| \geq \frac{1}{2} \cdot \frac{(1-\delta) \cdot m \cdot \ell}{m^{\beta/2}} = \Omega(m^{1-\beta/2} \cdot \ell)$. In particular, from (*i*) we get that there exists $i \in [\ell] \setminus D_1$ for which $S^{(i,1)} \cap \left(\rho^{-1}(*) \setminus \mathsf{live}(C^{(1),\rho})\right) \neq \emptyset$.

Overall, it follows that there exists a restriction $\overline{\rho} \in \Gamma_N$ with $\overline{\rho} = \rho \circ \sigma$, for an appropriate choice of $\rho \in \Gamma_{N_1}$, such that $\overline{\rho}$ fixes the message sent by Alice, but does not fix the value of $f^{\overline{\rho}}_{m,\ell,1}$. In particular, there exists a $z \in \{0,1\}^{n_1}$ that agrees with $\overline{\rho}$ for which $\Pi(z) \neq f^{\sigma}_{m,\ell,1}(z)$, which completes the proof. □

The proof of Lemma 4.7.4 is not much harder than the argument used in the base case, but it has a few technicalities that need to be handled.

Proof of Lemma 4.7.4. Let $\sigma \in \Gamma_{N,\delta}$ and $\Pi \in \mathsf{Prot}^{\sigma}_{s,d,r}$ be a pair realizing $\phi_{s,d,\ell}(m,r,\delta)$. In other words, Π solves the compression game of $f^{\sigma}_{m,\ell,r}$, and $\mathsf{cost}(\Pi) = \phi_{s,d,\ell}(m,r,\delta)$. Assume that $\Pi = (C^{(1)}, \ldots, C^{(r)}, g^{(1)}, \ldots, g^{(r-1)}, E)$, and $\mathsf{signature}(\Pi) = (n_1, s_1, t_1, \ldots, t_{r-1}, s_r)$, where $n = m + m \cdot \ell \cdot r$, $N = [n]$, $N_1 = N \setminus \sigma^{-1}(\{0,1\})$, and $n_1 = |N_1|$. For convenience, let $C^{(1)} = (C_1, \ldots, C_{s_1})$, where each C_i is a depth-d AC^0 circuit of size at most $n^{c_1} \leq n_1^{2c_1}$ (for large m), since $n_1 \geq (1-\delta) \cdot n$.

Notice that if $\mathsf{cost}(\Pi) \geq \ell \cdot m^{1-\beta}$ then the statement of Lemma 4.7.4 is true. Otherwise, from $\mathsf{cost}(\Pi) < \ell \cdot m^{1-\beta}$ we get that $s_1 < \ell \cdot m^{1-\beta}$, which allows us to proceed as in the

proof of Lemma 4.7.3. Let $p = m^{-\beta/2}$, and set $\gamma = \beta$. It follows from Lemma 4.7.8 that, with high probability,

$$\left|\mathsf{live}(C^{(1),\rho})\right| = O(m^{1-\beta} \cdot \ell). \tag{4.3}$$

Let D_j for $j \in [r]$ be the sets identifying the variables y fixed by σ. By assumption, $|D_j| \leq \delta \cdot \ell$ for every $j \in [r]$. From now on, whenever we consider a set $S^{(i,j)}$, we implicitly assume that $j \in [r]$ and $i \in [\ell] \setminus D_j$. This time we will also be concerned about how the action of ρ affects the more specific sets $S^{(i,j,u)}$, where $u \in [\log \ell]$. Observe that, with high probability (Proposition 2.0.1), for every (i, j, u), we have:

$$\left|S^{(i,j,u)} \cap \rho^{-1}(*)\right| \geq \frac{1}{2} \cdot \frac{m}{\log \ell} \cdot p = \frac{1}{2} \cdot \frac{m^{1-\beta/2}}{c_2 \log m} \geq m^{1-(3/4)\beta}, \tag{4.4}$$

for any sufficiently large m. We say that a set $S^{(i,j)}$ is *bad* with respect to $C^{(1),\rho}$ if $|S^{(i,j)} \cap \mathsf{live}(C^{(1),\rho})| \geq \frac{1}{2} \cdot m^{1-(3/4)\beta}$. Otherwise, the set is said to be *good*. It follows from Equation 4.3 that

$$\text{Number of bad sets } S^{(i,j)} \leq \frac{O(m^{1-\beta} \cdot \ell)}{(1/2) \cdot m^{1-(3/4)\beta}} = \frac{2\ell}{m^{\beta/4}} = o(\ell), \tag{4.5}$$

as $m \to \infty$. In particular, since $r = O(1)$ and β is a fixed constant, with high probability, for every $j \in [r]$ there are at most $\beta \cdot \ell$ sets $S^{(i,j)}$ that are bad with respect to $C^{(1),\rho}$. Finally, with high probability over ρ, we also get that, for every $j \in [\log \ell]$,

$$\left|M^{(j)} \cap \rho^{-1}(*)\right| > 0.$$

It follows using the probabilistic method that there exists a fixed restriction $\rho_1 \in \Gamma_{N_1}$ satisfying all these properties. Let $\rho_2 = \rho_1 \circ \sigma$ be the restriction obtained by combining ρ_1 and σ in the obvious way. Observe that $\rho_2 \colon N \to \{0, 1, *\}$. Fix arbitrarily all $*$-variables in ρ_2 corresponding to bad sets $S^{(i,j)}$. On every good set $S^{(i,j)}$, fix all $*$-variables intersecting $\mathsf{live}(C^{(1),\rho_1})$, and also fix additional variables in each set $S^{(i,j,u)}$ so that the new restriction ρ_3 satisfies $|\rho_3^{-1}(*) \cap S^{(i,j,u)}| = m^{1-\beta}$, for every appropriate triple (i, j, u). This is possible for any large enough m, since these sets are good. Further, we assume that the number of variables corresponding to each $S^{(i,j,u)}$ that are set to 1 is *even*, in order not to invert the parity inside each block, which will be important later in the proof. Let $f^{\rho_3}_{m,\ell,r} \colon \{0,1\}^{\rho_3^{-1}(*)} \to \{0, 1\}$ be the resulting function.

Given an input $\tilde{z} \in \{0,1\}^{\rho_3^{-1}(*)}$, write $\tilde{z} = (\tilde{x}, \{\tilde{y}^{(i,j)}\})$, and let $z = (x, \{y^{(i,j)}\}) \in \{0,1\}^n$ be the completion of $\tilde{z}$ that *agrees* with ρ_3, where this notion is defined in the natural way. Observe that $h(x)$ still depends on $\tilde{x}$. Now we set all remaining $*$-variables in M in a way that, for the new restriction $\overline{\sigma}\colon [N] \to \{0,1,*\}$, we have $h(\overline{\sigma}(M))$ pointing to a pair $(i,1)$ corresponding to a good set $S^{(i,1)}$. This is possible due to the properties of ρ_1. Observe that $C^{(1),\overline{\sigma}}$ computes a constant function (i.e., Alice's message $a^{(1)}$ has been fixed). Let $b^{(1)} \in \{0,1\}^{t_1}$ be the answer provided by Bob, which is also fixed.

Now let $\overline{\Pi} = (\overline{C}^{(1)}, \ldots, \overline{C}^{(r-1)}, \overline{g}^{(1)}, \ldots, \overline{g}^{(r-2)}, E)$ be a new protocol obtained by setting each $\overline{C}^{(i)}$ to be $C^{(i+1)}$ with its input corresponding to the first message sent by Bob fixed to $b^{(1)}$, and $\overline{g}^{(i)} = g^{(i+1)}$, for every appropriate i. If we also rename the input variables in $f^{\overline{\sigma}}_{m,\ell,r}$ and in the functions and circuits from $\overline{\Pi}$, truncating irrelevant variables appropriately (recall the definition of the original function as a pointer jumping function), we obtain a restriction $\sigma'\colon \{0,1\}^{N'} \to \{0,1\}$, where $n' = |N'| = m' + m' \cdot \ell \cdot r'$, $m' = m^{1-\beta}$, $r' = r-1$, $\sigma' \in \Gamma_{N',\delta'}$, $\delta' = \delta + \beta$, and the resulting protocol $\Pi' \in \mathsf{Prot}^{\sigma'}_{s,d,r'}$. Crucially, Π' is a protocol solving the compression game of $f^{\sigma'}_{m',\ell,r'}$ in r' rounds, which implies that $\mathsf{cost}(\Pi) \geq \mathsf{cost}(\Pi') \geq \phi_{s,d,\ell}(m',r',\delta') = \phi_{s,d,\ell}(m^{1-\beta}, r-1, \delta+\beta)$, completing the proof of Lemma 4.7.4. □

4.8 Open problems and further research directions

Our results and techniques raise a number of interesting questions, which we discuss more carefully below.

The power of interaction in two-party $\mathsf{AC}^0[p]$-compression games. Observe that the approach to obtain communication lower bounds for $\mathsf{AC}^0[p]$ games employed in the proof of Theorem 4.1.1 is insensitive to the number of rounds of the protocol. On the other hand, our round separation result (Theorem 4.1.6) holds with respect to AC^0 circuits only. Consequently, a natural question is whether a strong round separation theorem is true for $\mathsf{AC}^0[p]$ games. We conjecture that this is the case, and that a hard function can be obtained via a similar construction that uses MOD_q instead of parity.

Randomized $\mathsf{AC}^0[p]$-compression games. While we have obtained essentially optimal

lower bounds for deterministic two-party $\mathsf{AC}^0[p]$-compression games, the situation is less clear with respect to randomized protocols. Modulo logarithmic factors, there is a quadratic gap between our upper and lower bounds for MOD_q and $\mathsf{Majority}$ (Theorem 4.1.3). On the other hand, it is known that the communication cost of these games is $n/\log^{\Theta(d)} n$ for randomized AC^0_d-compression games (Chattopadhyay and Santhanam [45]). We are unable to obtain better lower bounds here because our approach does not seem to tolerate the initial error probability from the protocol, as it relies on the low error regime of the polynomial approximation method.

Extending circuit lower bounds to incompressibility results. The results presented in this chapter and in [45] show that recent extensions of the random restriction method and the polynomial approximation method can provide optimal incompressibility results. However, our construction from Section 4.6 implies that not every technique can be extended in this sense. Which other techniques and results from circuit complexity can be strengthened to compressibility lower bounds?

Understanding the structure of Boolean circuits. Our results shed more light into the computation of Boolean functions such as MOD_q using $\mathsf{AC}^0[p]$ circuits, as we are able to obtain information about each layer of the circuit. Similar developments appear for instance in Tarui [184], Rudich and Berman [163], and Borodin [38]. We believe that results of this form can provide important insights in algorithms and computational complexity, and it would be very interesting to see further advances in this direction.

4.9 Auxiliary results

The degree lower bound in the low-error regime. In this section we describe the proof of the degree lower bound for $\mathbb{F}_p$-polynomials approximating MOD_q in the low error regime. Recall that we use MOD^n_q to denote the MOD_q function over n input variables, and that a polynomial $Q \in \mathbb{F}_p[x_1, \ldots, x_n]$ $\varepsilon(n)$-approximates a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ under the uniform distribution if

$$\Pr_{x \sim \{0,1\}^n}[Q(x) = f(x)] \geq 1 - \varepsilon(n),$$

where x is viewed as an element of $\mathbb{F}_p^n$ or $\{0,1\}^n$, depending on the context.

Proposition 4.9.1 ([157, 177], folklore)**.** *Let $p, q \geq 2$ be distinct primes. There exist fixed constants $\delta > 0$ and $n_0 \in \mathbb{N}$ for which the following holds. For every $n \geq n_0$ and $\varepsilon(n) \in [2^{-n}, 1/10q]$, any polynomial $P \in \mathbb{F}_p[x_1, \ldots, x_n]$ that ε-approximates the MOD_q^n function with respect to the uniform distribution has degree at least $\delta \cdot \sqrt{n \cdot \log(1/\varepsilon)}$.*

The proofs that appear in the literature are concerned with large values of ε, and our goal here is to discuss the extension of the degree lower bound to very small ε, as stated in Proposition 4.9.1. For this reason, we will focus on the case where $q = 2$ and $p > 2$, which is slightly simpler. We start with the following lemma.

Lemma 4.9.2. *For a prime $p > 2$, let $P \in \mathbb{F}_p[x_1, \ldots, x_n]$ be a degree-d polynomial that $\varepsilon(n)$-approximates MOD_2^n over the uniform distribution. Then there exists a polynomial $Q \in \mathbb{F}_p[y_1, \ldots, y_n]$ of degree at most d and a set $S \subseteq \{-1,1\}^n \subseteq \mathbb{F}_p^n$ with $|S| \geq (1-\varepsilon)2^n$ such that*

$$\forall y \in S, \quad Q(y) = \prod_{i=1}^{n} y_i.$$

Proof. Let $T \subseteq \{0,1\}^n \subseteq \mathbb{F}_p^n$ be a set of size at least $(1-\varepsilon)2^n$ such that

$$\forall x \in T, \quad P(x) = \mathsf{MOD}_2^n(x).$$

Consider the map $\gamma\colon \{-1,1\} \to \{0,1\}$ computed by the $\mathbb{F}_p$-polynomial $\gamma(y) \stackrel{\text{def}}{=} (1-y)2^{-1}$. Observe that $\gamma(-1) = 1$ and $\gamma(1) = 0$. Let $Q(y_1, \ldots, y_n)$ be a polynomial in $\mathbb{F}_p[y_1, \ldots, y_n]$ with $Q(y) \stackrel{\text{def}}{=} 2P(\gamma(y_1), \ldots, \gamma(y_n)) - 1$, and let

$$S \stackrel{\text{def}}{=} \{y \in \{-1,1\}^n \mid (y_1, \ldots, y_n) = (\gamma^{-1}(x_1), \ldots, \gamma^{-1}(x_n)), \text{ where } x \in T\}.$$

Then, using the definition of P, Q, S, T, and γ, it is not hard to see that

$$\forall y \in S, \quad Q(y) = \prod_{i=1}^{n} y_i.$$

Finally, observe that $|S| = |T|$ and $\mathsf{deg}(Q) \leq \mathsf{deg}(P)$, which completes the proof of the lemma. □

The next lemma shows that polynomials with this property can be very useful when computing functions defined over $S \subset \mathbb{F}_p^n$.

Lemma 4.9.3. *Let $\mathbb{F}$ be a finite field, and $a, b \in \mathbb{F}$ be distinct non-zero elements. Assume that $Q \in \mathbb{F}[x_1, \ldots, x_n]$ is a degree-d polynomial, and $S \subseteq \{a, b\}^n$ is a set such that*

$$\forall x \in S, \quad Q(x) = \prod_{i=1}^{n} x_i.$$

Then, for every function $f \colon S \to \mathbb{F}$, there is a polynomial $Q_f \in \mathbb{F}[x_1, \ldots, x_n]$ with degree at most $(n+d)/2$ such that

$$\forall x \in S, \quad Q_f(x) = f(x).$$

Proof. Fix a function $f \colon S \to \mathbb{F}$, and let P_f be a multilinear polynomial such that, for all $x \in S$, $P_f(x) = f(x)$. For instance, since a and b are distinct elements of $\mathbb{F}$, we can take

$$P_f(x) \stackrel{\text{def}}{=} \sum_{x \in S} f(x) \cdot \left(\prod_{i : x_i = a} (b - x_i)(b-a)^{-1} \right) \left(\prod_{i : x_i = b} (a - x_i)(a-b)^{-1} \right).$$

Now consider any monomial $M(x) \stackrel{\text{def}}{=} \prod_{i \in I} x_i$, where $I \subseteq [n]$. Since a and b are non-zero, for any $y \in S \subseteq \{a, b\}^n$, we have

$$\begin{aligned} \prod_{i \in I} y_i &= \left(\prod_{i \in [n]} y_i \right) \left(\prod_{i \notin I} y_i^{-1} \right) \\ &= Q(y) \cdot \left(\prod_{i \notin I} a^{-1}(b - y_i)(b-a)^{-1} + b^{-1}(a - y_i)(a-b)^{-1} \right), \end{aligned}$$

where Q is the polynomial granted by the statement of the lemma. Therefore, each monomial in P_f defined over a subset $I \subseteq [n]$ can be replaced by a monomial of degree at most $\min(|I|, d + n - |I|) \leq (n+d)/2$, in the sense that the new polynomial is still correct on every input in S. Consequently, there exists a polynomial Q_f for f with degree at most $(n+d)/2$, as claimed by the lemma. □

In other words, if d is small, there exist polynomials of degree much smaller than n for all functions with domain S and codomain $\mathbb{F}$. This is impossible for large sets S, via a simple counting argument. In order to formalize this argument and obtain good parameters, we rely on a certain lower bound for the binomial distribution. The next lemma follows from more general results presented in Feller [65]. We follow closely the exposition in Matoušek and Vondrák [135].

Lemma 4.9.4. *For an even integer $n \in \mathbb{N}$, consider independent random variables $X_1, \ldots, X_n$, where each X_i attains values 0 and 1, each with probability 1/2. Let $X \stackrel{\text{def}}{=} \sum_{i \in [n]} X_i$. Then, for any integer $t \in [0, n/8]$,*

$$\Pr\left[X \geq \frac{n}{2} + t\right] \geq \frac{1}{15} \cdot e^{-16t^2/n}.$$

Proof. For convenience, let $n = 2m$. Then,

$$\begin{aligned}
\Pr[X \geq m + t] &= 2^{-2m} \sum_{j=t}^{m} \binom{2m}{m+j} \\
&\geq 2^{-2m} \sum_{j=t}^{2t-1} \binom{2m}{m+j} \\
&= 2^{-2m} \sum_{j=t}^{2t-1} \binom{2m}{m} \frac{m}{m+j} \cdot \frac{m-1}{m+j-1} \cdots \frac{m-j+1}{m+1} \\
&\geq \frac{1}{2\sqrt{m}} \sum_{j=t}^{2t-1} \prod_{i=1}^{j} \left(1 - \frac{j}{m+i}\right) \quad \left(\text{since } \binom{2m}{m} \geq 2^{2m}/(2\sqrt{m})\right) \\
&\geq \frac{t}{2\sqrt{m}} \left(1 - \frac{2t}{m}\right)^{2t} \\
&\geq \frac{t}{2\sqrt{m}} \cdot e^{-8t^2/m} \quad (\text{since } 1 - x \geq e^{-2x} \text{ for } 0 \leq x \leq 1/2).
\end{aligned}$$

The lemma now follows depending on the value of t. Observe that if $t \geq \frac{1}{4}\sqrt{m}$ then the last expression is lower bounded by $\frac{1}{8}e^{-16t^2/n}$. On the other hand, for $0 \leq t < \frac{1}{4}\sqrt{m}$, we get that $\Pr[X \geq m + t] \geq \Pr[X \geq m + \frac{1}{4}\sqrt{m}] \geq \frac{1}{8}e^{-1/2} \geq \frac{1}{15}$, which completes the proof. □

Finally, we combine these lemmas in order to prove Proposition 4.9.1 for primes $q = 2$ and $p > 2$.

Proof. Let $P \in \mathbb{F}_p[x_1, \ldots, x_n]$ be a degree-d polynomial that $\varepsilon(n)$-approximates the MOD_2^n function over the uniform distribution. Assume without loss of generality that n is even, since otherwise we can obtain a polynomial $Q \in \mathbb{F}_p[x_1, \ldots, x_{n+1}]$ with degree at most $2d$ that $\varepsilon(n)$-approximates MOD_2^{n+1} with respect to $\{0,1\}^{n+1}$ (i.e., apply P to the first n variables, then compose with the appropriate function over two input variables).

It follows from Lemmas 4.9.2 and 4.9.3 that there exists a set $S \subseteq \{-1,1\}^n \subseteq \mathbb{F}_p^n$ of size $(1-\varepsilon)2^n$ such that, for every function $f\colon S \to \mathbb{F}_p$, there exists a polynomial $Q_f \in \mathbb{F}_p[x_1,\ldots,x_n]$ of degree at most $d' \stackrel{\text{def}}{=} (n+d)/2$ that agrees with f over S.

Let $\mathcal{F}$ be the set of such functions. Clearly, $|\mathcal{F}| = |\mathbb{F}_p|^{|S|}$. On the other hand, since $S \subseteq \{-1,1\}^n$, we can assume that each polynomial Q_f is multilinear. The number of such polynomials with degree at most d' is upper bounded by $|\mathbb{F}_p|^M$, where $M \stackrel{\text{def}}{=} \sum_{i=0}^{d'} \binom{n}{i}$. Therefore, $|\mathbb{F}_p|^{|S|} \leq |\mathcal{F}| \leq |\mathbb{F}_p|^M$, and we get that

$$\sum_{i=0}^{(n+d)/2} \binom{n}{i} \geq (1-\varepsilon)\cdot 2^n. \tag{4.6}$$

We use this inequality to lower bound d in terms of n and ε. First, Equation 4.6 can be rewritten as

$$2^{-n} \cdot \sum_{i>(n+d)/2} \binom{n}{i} \leq \varepsilon. \tag{4.7}$$

On the other hand, it follows from Lemma 4.9.4 that, for any $d \in [0, n/8]$,

$$\frac{1}{15}\cdot \exp\left(-\frac{16}{n}\cdot\left(\frac{d}{2}+1\right)^2\right) \leq \Pr\left[X > \frac{n}{2}+\frac{d}{2}\right] = 2^{-n}\cdot \sum_{i>(n+d)/2}\binom{n}{i}. \tag{4.8}$$

Therefore, we obtain from Equations 4.7 and 4.8 that $d = \Omega(\sqrt{n\cdot\log(1/\varepsilon)})$ for any $\varepsilon(n) \in [2^{-n}, 1/20]$, which completes the proof. □

Improved approximation of $\mathsf{AC}^0[p]$ circuits by polynomials. For convenience of the reader, we describe in this section how to approximate Boolean circuits by bounded-degree polynomials in the low-error regime. We assume the following classic result, obtained in slightly different forms by Razborov [157] and Smolensky [177].

Proposition 4.9.5 ([157], [177])**.** *Let p be a fixed prime. There exists a constant $\beta = \beta(p) \in \mathbb{N}$ such that, for every $d = d(n) \geq 1$ and $s = s(n) \geq 1$, any $\mathsf{AC}^0_d[p](s(n))$ circuit admits an $1/(6s)$-error probabilistic polynomial $\mathbf{Q}(x_1,\ldots,x_n) \in \mathbb{F}_p[x_1,\ldots,x_n]$ of degree at most $(\beta\cdot\log\max\{s,2\})^d$.*

We are now ready to describe the proof of the degree upper bound obtained by Kopparty and Srinivasan [123], which allows us to obtain better bounds when the error is sufficiently small.

Proposition 4.9.6 ([123])**.** *Let p be a fixed prime. There exists a constant $\alpha = \alpha(p) \in \mathbb{N}$ such that, for every $\delta \in (0, 1/2)$ and $d(n) \geq 2$, any $\mathsf{AC}^0_d[p](s(n))$ circuit C admits a δ-error probabilistic polynomial $\mathbf{Q}(x_1, \ldots, x_n) \in \mathbb{F}_p[x_1, \ldots, x_n]$ of degree at most $(\alpha \cdot \log s)^{d-1} \cdot \log(1/\delta)$. In particular, it follows that for any distribution $\mathcal{D}$ over $\{0,1\}^n$, C is δ-approximated with respect to $\mathcal{D}$ by a polynomial of degree at most $(\alpha \cdot \log s)^{d-1} \cdot \log(1/\delta)$.*

Proof. Let C be an $\mathsf{AC}^0[p]$ circuit of size s and depth $d \geq 2$. Further, let g be the top gate of C, and assume that this gate is fed by $t \leq s$ input wires $y_1, \ldots, y_t$, where each $y_j = g_j(x_1, \ldots, x_n)$. Observe that the corresponding Boolean function over inputs $x_1, \ldots, x_n$ at each gate g_j is computed by a circuit of size at most s and depth at most $d-1$, while $g = g(y_1, \ldots, y_t)$ is computed by a circuit of size one. Let $\varepsilon \stackrel{\text{def}}{=} 1/(6s)$. Then, Proposition 4.9.5 guarantees the existence of probabilistic polynomials $\mathbf{Q}_j(x_1, \ldots, x_n)$ which compute the corresponding functions g_j with error at most ε, where $\mathsf{deg}(\mathbf{Q}_j) \leq (\beta \cdot \log s)^{d-1}$. Similarly, since g is computed by a single gate, there exists a probabilistic polynomial $\mathbf{Q}_g(y_1, \ldots, y_t)$ that computes g with error at most $1/6$, where $\mathsf{deg}(\mathbf{Q}_g) \leq \beta$. By composing these polynomials and applying a union bound, it follows that there exists a probabilistic polynomial $\mathbf{P}(\vec{x}) \stackrel{\text{def}}{=} \mathbf{Q}_g(\mathbf{Q}_1(\vec{x}), \ldots, \mathbf{Q}_t(\vec{x}))$ with $\mathsf{deg}(\mathbf{P}) \leq (\gamma \cdot \log s)^{d-1}$ that computes C with error at most $1/3$, where $\gamma = \gamma(p)$ is a fixed constant. Further, by raising this polynomial to $p-1$ and applying Fermat's little theorem, we can assume without loss of generality that its output is always Boolean. Since $d \geq 2$, the degree becomes at most $(\gamma' \cdot \log s)^{d-1}$, where $\gamma' \leq p \cdot \gamma$.

Now let $k = c \cdot \log(1/\delta)$, for a sufficiently large constant c. Consider the probabilistic polynomial $\mathbf{M}(\vec{x}) \stackrel{\text{def}}{=} \mathsf{M}(\mathbf{P}_1(\vec{x}), \ldots, \mathbf{P}_k(\vec{x}))$, where M is a degree k polynomial that computes $\mathsf{Majority}_k$ exactly, and each $\mathbf{P}_i$ is an independent copy of $\mathbf{P}$. It follows from Proposition 2.0.1 that $\mathbf{M}$ is a probabilistic polynomial of degree at most $(\alpha \cdot \log s)^{d-1} \cdot \log(1/\delta)$ that computes C with error at most δ, where $\alpha = \alpha(\gamma', c) = \alpha(p)$ is an appropriate constant. □

Part II

Negations in Learning Theory and Cryptography

Chapter 5

Learning circuits with negations

5.1 Background, results, and organization

Recall that a *monotone* Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is one that satisfies $f(x) \leq f(y)$ whenever $x \preceq y$, where $\preceq$ denotes the bitwise partial order on $\{0,1\}^n$. The structural and combinatorial properties of monotone Boolean functions have been intensively studied for many decades, see e.g. [125] for an in-depth survey. Many famous results in circuit complexity deal with monotone functions, including celebrated lower bounds on monotone circuit size and monotone formula size (see e.g. [153, 156] and numerous subsequent works).

Monotone functions are also of considerable interest in computational learning theory, in particular with respect to the model of learning under the uniform distribution. In an influential paper, Bshouty and Tamon [40] showed that any monotone Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ can be learned from uniform random examples to error ε in time $n^{O(\sqrt{n}/\varepsilon)}$. They also gave a lower bound, showing that no algorithm running in time 2^{cn} for any $c < 1$ can learn arbitrary monotone functions to accuracy $\varepsilon = 1/(\sqrt{n}\log n)$. (Many other works in learning theory such as [17, 116, 31, 14, 172, 144, 145] deal with learning monotone functions from a range of different perspectives and learning models, but we limit our focus in this chapter to learning to high accuracy with respect to the uniform distribution.)

5.1.1 Beyond monotonicity: inversion complexity and alternations

Given the importance of monotone functions in complexity theory and learning theory, it is natural to consider various generalizations of monotonicity. One such generalization arises from the simple observation that monotone Boolean functions are precisely the functions computed by *monotone Boolean circuits*, i.e. circuits which have only AND and OR gates but no negations. Given this, an obvious generalization of monotonicity is obtained by considering functions computed by Boolean circuits that have a small number of negation gates. The *inversion complexity* of $f\colon \{0,1\}^n \to \{0,1\}$, denoted $I(f)$, is defined to be the minimum number of negation gates in any AND/OR/NOT circuit (with access to constant inputs 0/1) that computes f. We write $\mathcal{C}_t^n$ to denote the class of n-variable Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ that have $I(f) \leq t$.

Another generalization of monotonicity is obtained by starting from an alternate characterization of monotone Boolean functions. A function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone if and only if the value of f "flips" from 0 to 1 at most once as the input x ascends any chain in $\{0,1\}^n$ from 0^n to 1^n. (Recall that a chain of length ℓ is an increasing sequence $(x^1, \ldots, x^\ell)$ of vectors in $\{0,1\}^n$, i.e. for every $j \in \{1, \ldots, \ell-1\}$ we have $x^j \prec x^{j+1}$.) Thus, it is natural to consider a generalization of monotonicity that allows more than one such "flip" to occur. We make this precise with the following notation and terminology: given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and a chain $X = (x^1, \ldots, x^\ell)$, a position $j \in [\ell - 1]$ is said to be *alternating* with respect to f if $f(x^j) \neq f(x^{j+1})$. We write $A(f, X) \subseteq [\ell - 1]$ to denote the set of alternating positions in X with respect to f, and we let $a(f, X) = |A(f, X)|$ denote its size. We write $a(f)$ to denote the maximum of $a(f, X)$ taken over all chains X in $\{0,1\}^n$, and we say that $f\colon \{0,1\}^n \to \{0,1\}$ is *k-alternating* if $a(f) \leq k$.

A celebrated result of A. A. Markov [132] gives a tight quantitative connection between the inversion and alternation complexities defined above:

Markov's Theorem. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function which is not identically 0. Then (i) if $f(0^n) = 0$, then $I(f) = \lceil \log(a(f) + 1) \rceil - 1$; and (ii) if $f(0^n) = 1$, then $I(f) = \lceil \log(a(f) + 2) \rceil - 1$.*

This robustness motivates the study of circuits which contain few negation gates, and

indeed such circuits have been studied in complexity theory. For instance, Amano and Maruoka [15] have given bounds on the computational power of such circuits, showing that circuits for the clique function which contain fewer than $\frac{1}{6}\log\log n$ many negation gates must have superpolynomial size. More recently, Rossman [162] proved that there exists an explicit monotone function that cannot be computed by fan-in two circuits of logarithmic depth containing less than $\left(\frac{1}{2}-\varepsilon\right)\log n$ negations. Other works have studied the effect of limiting the number of negation gates in formulas [138], bounded-depth circuits [165, 181], and non-deterministic circuits [139]. In the present work, we study circuits with few negations from the vantage point of computational learning theory, giving both positive and negative results.

5.1.2 Our results

We begin by studying the structural properties of functions that are computed or approximated by circuits with few negation gates. In Section 5.2 we establish the following extension of Markov's theorem:

Theorem 5.1.1. *Let f be a k-alternating Boolean function. Then $f(x) = h(m_1(x), \ldots, m_k(x))$, where each $m_i(x)$ is monotone and h is either the parity function or its negation. Conversely, any function of this form is k-alternating.*

Theorem 5.1.1 along with Markov's theorem yields the following characterization of $\mathcal{C}_t^n$:

Corollary 5.1.2. *Every $f \in \mathcal{C}_t^n$ can be expressed as $f = h(m_1, \ldots, m_T)$ where h is either PAR_T or its negation, each $m_i \colon \{0,1\}^n \to \{0,1\}$ is monotone, and $T = O(2^t)$.*

A well-known consequence of Markov's theorem is that every Boolean function is exactly computed by a circuit which has only $\log n$ negation gates, and as we shall see an easy argument shows that every Boolean function is 0.01-approximated by a circuit with $\frac{1}{2}\log n + O(1)$ negations. In Section 5.2 we note that no significant savings are possible over this upper bound:

Theorem 5.1.3. *For almost every function $f \colon \{0,1\}^n \to \{0,1\}$, any Boolean circuit C that 0.01-approximates f must contain $\frac{1}{2}\log n - O(1)$ negations.*

We then turn to our main topic of investigation, the uniform-distribution learnability of circuits with few negations. We use our new extension of Markov's theorem, Theorem 5.1.1, to obtain a generalization of the Fourier-based uniform-distribution learning algorithm of Bshouty and Tamon [40] for monotone circuits:

Theorem 5.1.4. *There is a uniform-distribution learning algorithm which learns any unknown $f \in \mathcal{C}_t^n$ from random examples to error ε in time $n^{O(2^t\sqrt{n}/\varepsilon)}$.*

We observe that many natural functions are indeed computed by circuits with few negations. As an example, consider the property of undirected graphs that is satisfied by an n-vertex graph G if and only if G contains a triangle but does not contain a cycle of size $\log n$. Clearly, this property is non-monotone. However, it is easy to see that it can be represented by a Boolean function $f\colon \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ that is computed by a circuit with a single negation. Our positive result implies that learning such properties does not take much more time than learning monotone properties.[1]

Theorem 5.1.4 immediately leads to the following question: can an even faster learning algorithm be given for circuits with t negations, or is the running time of Theorem 5.1.4 essentially the best possible? Interestingly, prior to our work a matching lower bound for Theorem 5.1.4 was not known even for the special case of monotone functions (corresponding to $t = 0$). As mentioned earlier, Bshouty and Tamon proved that to achieve accuracy $\varepsilon = 1/(\sqrt{n}\log n)$ any learning algorithm needs time $\omega(2^{cn})$ for any $c < 1$ (see Fact 5.4.11 for a slight sharpening of this statement). For larger values of ε, though, the strongest previous lower bound was due to Blum, Burch and Langford [31]. Their Theorem 10 implies that any membership-query algorithm that learns monotone functions to error $\varepsilon < \frac{1}{2} - c$ (for any $c > 0$) must run in time $2^{\Omega(\sqrt{n})}$ (in fact, must make at least this many membership queries). However, this lower bound does not differentiate between the number of membership queries required to learn to high accuracy versus "moderate" accuracy – say, $\varepsilon = 1/n^{1/10}$ versus $\varepsilon = 1/10$. Thus the following question was unanswered prior to the current work: what is

[1] In contrast to the robustness we show in the learning setting, there are natural computational problems whose complexity changes drastically with the addition of a single negation gate. For instance, checking if a monotone circuit is non-constant is trivial. Nevertheless, it is possible to prove that the same computational problem for circuits with a single negation gate admits polynomial time algorithms if and only if $\mathsf{P} = \mathsf{NP}$.

the best lower bound that can be given, both as a function of n and ε, on the complexity of learning monotone functions to accuracy ε?

We give a fairly complete answer to this question, providing a lower bound as a function of n, ε and t on the complexity of learning circuits with t negations. Our lower bound essentially matches the upper bound of Theorem 5.1.4, and is thus simultaneously essentially optimal in all three parameters n, ε and t for a wide range of settings of ε and t. Our lower bound result is the following:

Theorem 5.1.5. *For any $t \leq \frac{1}{28} \log n$ and any $\varepsilon \in [1/n^{1/12}, 1/2-c]$, $c > 0$, any membership-query algorithm that learns any unknown function $f \in \mathcal{C}_t^n$ to error ε must make $2^{\Omega(2^t\sqrt{n}/\varepsilon)}$ membership queries.*

We note that while our algorithm uses only uniform random examples, our lower bound holds even for the stronger model in which the learning algorithm is allowed to make arbitrary membership queries on points of its choosing.

Theorem 5.1.5 is proved using tools from the study of hardness amplification. The proof involves a few steps. We start with a strong lower bound for the task of learning to high accuracy the class of balanced monotone Boolean functions (reminiscent of the lower bound obtained by Bshouty and Tamon). Then we combine hardness amplification techniques and results on the noise sensitivity of monotone functions in order to get stronger and more general lower bounds for learning monotone Boolean functions to moderate accuracy. Finally, we use hardness amplification once more to lift this result into a lower bound for learning circuits with few negations to moderate accuracy. An ingredient employed in this last stage is to use a k-alternating combining function which "behaves like" the parity function on (roughly) k^2 variables; this is crucial in order for us to obtain our essentially optimal final lower bound of $2^{\Omega(2^t\sqrt{n}/\varepsilon)}$ for circuits with t negations. These results are discussed in more detail in Section 5.4.

5.2 Structural results

5.2.1 An extension of Markov's theorem

We begin with the proof of our new extension of Markov's theorem. For any $A \subseteq \{0,1\}^n$ let $\mathbf{1}[A] : \{0,1\}^n \to \{0,1\}$ be the characteristic function of A. For $f\colon \{0,1\}^n \to \{0,1\}$ and $x \in \{0,1\}^n$, we write $a_f(x)$ to denote

$$a_f(x) \stackrel{\text{def}}{=} \max\{a(f,X) : X \text{ is a chain that starts at } x\},$$

and note that $a(f) = \max_{x\in\{0,1\}^n}\{a_f(x)\} = a_f(0^n)$. For $0 \le \ell \le a(f)$ let us write S^f_ℓ to denote $S^f_\ell \stackrel{\text{def}}{=} \{x \in \{0,1\}^n : a_f(x) = \ell\}$, and let $T^f_\ell \stackrel{\text{def}}{=} S^f_0 \cup \cdots \cup S^f_\ell$. We note that $S^f_1, \ldots, S^f_{a(f)}$ partition the set of all inputs: $S^f_i \cap S^f_j = \emptyset$ for all $i \neq j$, and $T^f_{a(f)} = S^f_1 \cup \cdots \cup S^f_{a(f)} = \{0,1\}^n$.

We will need the following simple observation:

Observation 5.2.1. *Fix any f and any $x \in \{0,1\}^n$. If $x \in S^f_\ell$ and $y \succ x$ then $y \in S^f_{\ell'}$ for some $\ell' \le \ell$. Furthermore, if $f(y) \neq f(x)$ then $\ell' < \ell$.*

We prove Theorem 5.1.1 next, restated below.

Theorem. *Fix $f\colon \{0,1\}^n \to \{0,1\}$ and let $k \stackrel{\text{def}}{=} a(f)$. Then $f = h\big(\mathbf{1}[T^f_0], \ldots, \mathbf{1}[T^f_{k-1}]\big)$, where*

(i) the functions $\mathbf{1}[T^f_\ell]$ are monotone for all $0 \le \ell \le k$,

(ii) $h : \{0,1\}^k \to \{0,1\}$ is PAR_k if $f(0^n) = 0$ and $\neg\,\mathsf{PAR}_k$ if $f(0^n) = 1$,

and $\mathsf{PAR}_k(x) = x_1 \oplus \cdots \oplus x_k$ is the parity function on k variables. Conversely, given monotone Boolean functions $m_1, \ldots, m_k$, any Boolean function of the form $h(m_1, \ldots, m_k)$ is k-alternating.

Proof. Claim (i) follows immediately from Observation 5.2.1 above. The proof of (ii) is by induction on k. In the base case $k = 0$, we have that f is a constant function and the claim is immediate.

For the inductive step, suppose that the claim holds for all functions f' that have $a(f') \le k-1$. We define $f'\colon \{0,1\}^n \to \{0,1\}$ as $f' = f \oplus \mathbf{1}[S^f_k]$. Observation 5.2.1

implies that $S^{f'}_\ell = S^f_\ell$ for all $0 \le \ell \le k-2$ and $S^{f'}_{k-1} = S^f_{k-1} \cup S^f_k$, and in particular, $a(f) = k-1$. Therefore we may apply the inductive hypothesis to f' and express it as $f' = h'\big(\mathbf{1}\big[T^{f'}_0\big], \ldots, \mathbf{1}\big[T^{f'}_{k-2}\big]\big)$. Since $T^{f'}_\ell = T^f_\ell$ for $0 \le \ell \le k-2$, we may use this along with the fact that $\mathbf{1}\big[S^f_k\big] = \neg\, \mathbf{1}\big[T^f_{k-1}\big]$ to get:

$$f = f' \oplus \mathbf{1}\big[S^f_k\big] = h'\big(\mathbf{1}\big[T^{f'}_0\big], \ldots, \mathbf{1}\big[T^{f'}_{k-2}\big]\big) \oplus \neg\, \mathbf{1}\big[T^f_{k-1}\big] = h'\big(\mathbf{1}\big[T^f_0\big], \ldots, \mathbf{1}\big[T^f_{k-2}\big]\big) \oplus \neg\, \mathbf{1}\big[T^f_{k-1}\big]$$

and the inductive hypothesis holds (note that $0^n \in S^f_k$).

The converse is easily verified by observing that any chain in $\{0,1\}^n$ can induce at most $k+1$ possible vectors of values for $(m_1, \ldots, m_k)$ because of their monotonicity. □

Theorem 5.1.1 along with Markov's theorem immediately yield Corollary 5.1.2:

Corollary. *Every $f \in \mathcal{C}^n_t$ can be expressed as $f = h(m_1, \ldots, m_T)$ where h is either PAR_T or its negation, each $m_i \colon \{0,1\}^n \to \{0,1\}$ is monotone, and $T = O(2^t)$.*

5.2.2 Approximation

As noted earlier, Markov's theorem implies that every n-variable Boolean function can be exactly computed by a circuit with (essentially) $\log n$ negations (since $a(f) \le n$ for all f). If we set a less ambitious goal of *approximating* Boolean functions (say, having a circuit correctly compute f on a $1-\varepsilon$ fraction of all 2^n inputs), can significantly fewer negations suffice?

We first observe that every Boolean function f is ε-close (with respect to the uniform distribution) to a function f' that has $a(f') \le O(\sqrt{n \log 1/\varepsilon})$. The function f' is obtained from f simply by setting $f'(x) = 0$ for all inputs x that have Hamming weight outside of $[n/2 - O(\sqrt{n \log 1/\varepsilon}), n/2 + O(\sqrt{n \log 1/\varepsilon})]$; a standard Chernoff bound implies that f and f' disagree on at most $\varepsilon 2^n$ inputs. Markov's theorem then implies that the inversion complexity $I(f')$ is at most $\frac{1}{2}(\log n + \log\log \frac{1}{\varepsilon}) + O(1)$. Thus, every Boolean function can be approximated to high accuracy by a circuit with only $\frac{1}{2}\log n + O(1)$ negations.

We now show that this upper bound is essentially optimal: for almost every Boolean function, any 0.01-approximating circuit must contain at least $\frac{1}{2}\log n - O(1)$ negations. To prove this, recall the definition of *total influence* of a Boolean function, presented in Chapter

2. The total influence of f is easily seen to equal αn, where $\alpha \in [0,1]$ is the fraction of all edges $e = (x, x')$ in the Boolean hypercube that are bichromatic, i.e. have $f(x) \neq f(x')$. In Section 5.5 we prove the following lemma:

Lemma 5.2.2. *Suppose* $f\colon \{0,1\}^n \to \{0,1\}$ *is such that* $\mathrm{Inf}[f] = \Omega(n)$. *Then* $a(f) = \Omega(\sqrt{n})$.

It is easy to show that a random function has influence $\frac{n}{2}(1 - o(1))$ with probability $1 - 2^{-n}$. Given this, Claim 5.2.2, together with the elementary fact that if f' is ε-close to f then $|\mathrm{Inf}(f') - \mathrm{Inf}(f)| \leq 2\varepsilon n$, directly yields Theorem 5.1.3:

Theorem. *With probability* $1 - 2^{-n}$, *any* 0.01*-approximator* f' *for a random function* f *must have inversion complexity* $I(f') \geq \frac{1}{2}\log n - O(1)$.

Remark 2. *The results in this section (together with simple information-theoretic arguments showing that random functions are hard to learn) imply that one cannot expect to have a learning algorithm (even to constant accuracy) for the class* $\mathcal{C}^n_{\frac{1}{2}\log n + O(1)}$ *of circuits with* $\frac{1}{2}\log n + O(1)$ *negations in time significantly better than* 2^n. *As we shall see in* Section 5.3, *for any fixed* $\delta > 0$ *it is possible to learn* $\mathcal{C}^n_{(\frac{1}{2}-\delta)\log n}$ *to accuracy* $1 - \varepsilon$ *in time* $2^{\tilde{O}(n^{1-\delta})/\varepsilon}$.

5.3 A learning algorithm for non-monotone circuits

We sketch the learning algorithm and analysis of Bshouty and Tamon [40]. Using the results from Section 5.2, our Theorem 5.1.4 will follow easily from their approach. Our starting point is the simple observation that functions with good "Fourier concentration" can be learned to high accuracy under the uniform distribution simply by estimating all of the low-degree Fourier coefficients. This fact, established by Linial, Mansour and Nisan, is often referred to as the "Low-Degree Algorithm:"

Theorem 5.3.1 (Low-Degree Algorithm ([129]))**.** *Let* $\mathcal{C}$ *be a class of Boolean functions such that for* $\varepsilon > 0$ *and* $\tau = \tau(\varepsilon, n)$,

$$\sum_{|S| > \tau} \hat{f}(S)^2 \leq \varepsilon$$

for any $f \in \mathcal{C}$. *Then* $\mathcal{C}$ *can be learned from uniform random examples in time* $\mathrm{poly}(n^\tau, 1/\varepsilon)$.

Using the fact that every monotone function $f\colon \{0,1\}^n \to \{0,1\}$ has total influence $\mathrm{Inf}(f) \leq \sqrt{n}$, and the well-known Fourier expression $\mathrm{Inf}(f) = \sum_S \hat{f}(S) \cdot |S|^2$ for total influence, a simple application of Markov's inequality let Bshouty and Tamon show that every monotone function f has

$$\sum_{|S|>\sqrt{n}/\varepsilon} \hat{f}(S)^2 \leq \varepsilon.$$

Together with Theorem 5.3.1, this gives their learning result for monotone functions.

Armed with Corollary 5.1.2, it is straightforward to extend this to the class $\mathcal{C}_t^n$. Corollary 5.1.2 and a union bound immediately give that every $f \in \mathcal{C}_t^n$ has $\mathrm{Inf}(f) \leq O(2^t)\sqrt{n}$, so the Fourier expression for influence and Markov's inequality give that

$$\sum_{|S|>O(2^t)\sqrt{n}/\varepsilon} \hat{f}(S)^2 \leq \varepsilon$$

for $f \in \mathcal{C}_t^n$. Theorem 5.1.4 follows immediately using the Low-Degree Algorithm.

An immediate question is whether this upper bound on the complexity of learning $\mathcal{C}_t^n$ is optimal; we give an affirmative answer in the next section.

5.4 The complexity of learning non-monotone circuits

As noted in the introduction, we prove information-theoretic lower bounds against learning algorithms that make a limited number of membership queries. We start by establishing a new lower bound on the number of membership queries that are required to learn *monotone* functions to high accuracy, and then build on this to provide a lower bound for learning $\mathcal{C}_t^n$. Our query lower bounds are essentially tight, matching the upper bounds (which hold for learning from uniform random examples) up to logarithmic factors in the exponent.

We first state the results; the proofs are deferred to Subsection 5.4.1. We say that a Boolean function f is *balanced* if $\Pr_x[f(x) = 0] = \Pr_x[f(x) = 1] = 1/2$.

Theorem 5.4.1. *There exists a class $\mathcal{H}$ of balanced n-variable monotone Boolean functions such that for any $\varepsilon \in [\frac{1}{n^{1/6}}, 1/2 - c]$, $c > 0$, learning $\mathcal{H}_n$ to accuracy $1 - \varepsilon$ requires $2^{\Omega(\sqrt{n}/\varepsilon)}$ membership queries.*

This immediately implies the following corollary, which essentially closes the gap in our understanding of the hardness of learning monotone functions:

Corollary 5.4.2. *For any $\varepsilon = \Omega\big(1/n^{1/6}\big)$ bounded away from $1/2$, learning n-variable monotone functions to accuracy $1-\varepsilon$ requires $2^{\tilde{\Theta}(\sqrt{n})/\varepsilon}$ queries.*

Using this class $\mathcal{H}$ as a building block, we obtain the following hardness of learning result for the class of k-alternating functions:

Theorem 5.4.3. *For any function $k\colon \mathbb{N} \to \mathbb{N}$, there exists a class $\mathcal{H}^{(k)}$ of balanced $k = k(n)$-alternating n-variable Boolean functions such that, for any n sufficiently large and $\varepsilon > 0$ such that (i) $2 \le k < n^{1/14}$, and (ii) $k^{7/3}/n^{1/6} \le \varepsilon \le \frac{1}{2} - c$, learning $\mathcal{H}^{(k)}$ to accuracy $1-\varepsilon$ requires $2^{\Omega\left(k\sqrt{n}/\varepsilon\right)}$ membership queries.*

We note that the tradeoff between the ranges of k and ε that is captured by condition (ii) above seems to be inherent to our approach and not a mere artifact of the analysis; see Remark 4.

This theorem immediately yields the following results:

Corollary 5.4.4. *Learning the class of k-alternating functions to accuracy $1-\varepsilon$ in the uniform-distribution membership-query model requires $2^{\Omega\left(k\sqrt{n}/\varepsilon\right)}$ membership queries, for any $k = O\big(n^{1/28}\big)$ and $\varepsilon \in [1/n^{1/12}, \frac{1}{2} - c]$.*

Corollary 5.4.5. *For $t \le \frac{1}{28}\log n$, learning $\mathcal{C}_t^n$ to accuracy $1-\varepsilon$ requires $2^{\Omega\left(2^t\sqrt{n}/\varepsilon\right)}$ membership queries, for any $\varepsilon \in [2^{7t/3}/n^{1/6}, \frac{1}{2} - c]$.*

5.4.1 Proofs

We require the following standard notion of *composition* for two functions f and g:

Definition 5.4.6 (Composition)**.** *For $f\colon \{0,1\}^m \to \{0,1\}$ and $g\colon \{0,1\}^r \to \{0,1\}$, we denote by $g \otimes f$ the Boolean function on $n = mr$ inputs defined by*

$$(g \otimes f)(x) \stackrel{\text{def}}{=} g(\underbrace{f, \ldots, f}_{r})(x) = g(f(x_1, \ldots, x_m), \ldots, f(x_{(r-1)m+1}, \ldots, x_{rm}))$$

Similarly, for any $g\colon \{0,1\}^r \to \{0,1\}$ *and* $\mathcal{F}_m$ *a class of Boolean functions on* m *variables, we let*

$$g \otimes \mathcal{F}_m = \{\, g \otimes f \,:\; f \in \mathcal{F}_m \,\}$$

and $g \otimes \mathcal{F} = \{g \otimes \mathcal{F}_m\}_{m \geq 1}$.

Overview of the arguments. Our approach is based on hardness amplification. In order to get our lower bound against learning k-alternating functions, we (a) start from a lower bound ruling out very high-accuracy learning of *monotone* functions; (b) use a suitable monotone combining function to get an XOR-like hardness amplification, yielding a lower bound for learning (a subclass of) monotone functions to moderate accuracy; (c) repeat this approach on this subclass with a different (now k-alternating) combining function to obtain our final lower bound, for learning k-alternating functions to moderate accuracy.

$$\underbrace{\left[\begin{array}{c}\text{high-accuracy}\\ \text{monotone}\end{array}\right]}_{\text{(a)}} \xrightarrow[\text{monotone}]{\otimes\text{-like}} \underbrace{\left[\begin{array}{c}\text{moderate accuracy}\\ \text{monotone}\end{array}\right]}_{\text{(b)}} \xrightarrow[k\text{-alternating}]{\otimes\text{-like}} \underbrace{\left[\begin{array}{c}\text{moderate accuracy}\\ k\text{-alternating}\end{array}\right]}_{\text{(c)}} \tag{5.1}$$

In more detail, in both steps (b) and (c) the idea is to take as base functions the hard class from the previous step (respectively "monotone hard to learn to high accuracy", and "monotone hard to learn to moderate accuracy"), and compose them with a very noise-sensitive function in order to amplify hardness. Care must be taken to ensure that the combining function satisfies several necessary constraints (being monotone for (b) and k-alternating for (c), and being as sensitive as possible to the correct regime of noise in each case).

Useful tools. We begin by recalling a few notions and results that play a crucial role in our approach.

Definition 5.4.7 (Bias and expected bias)**.** *The* bias *of a Boolean function* $h\colon \{0,1\}^n \to \{0,1\}$ *is the quantity* $\mathrm{bias}(h) \stackrel{\text{def}}{=} \max(\Pr[\,h=1\,], \Pr[\,h=0\,])$*, while the* expected bias of h at δ *is defined as* $\mathrm{ExpBias}_\delta(h) \stackrel{\text{def}}{=} \mathrm{E}_\rho[\mathrm{bias}(h_\rho)]$*, where* ρ *is a random restriction on* k *coordinates where each coordinate is independently left free with probability* δ *and set to 0 or 1 with same probability* $(1-\delta)/2$.

Fact 5.4.8 (Proposition 4.0.11 from [146])**.** *For $\delta \in [0,1/2]$ and $f\colon \{0,1\}^n \to \{0,1\}$, we have*

$$\frac{1}{2} + \frac{1}{2}\,\mathrm{Stab}_{1-2\delta}(f) \leq \mathrm{ExpBias}_{2\delta}(f) \leq \frac{1}{2} + \frac{1}{2}\sqrt{\mathrm{Stab}_{1-2\delta}(f)}.$$

Building on Talagrand's probabilistic construction [182] of a class of functions that are sensitive to very small noise, Mossel and O'Donnell [140] gave the following noise stability upper bound. (We state below a slightly generalized version of their Theorem 3, which follows from their proof with some minor changes; see Subsection 5.5.2 for details of these changes.)

Theorem 5.4.9 (Theorem 3 of [140])**.** *There exists an absolute constant K and an infinite family of* balanced *monotone functions $g_r\colon \{0,1\}^r \to \{0,1\}$ such that $\mathrm{Stab}_{1-\tau/\sqrt{r}}(g_r) \leq 1 - K\tau$ holds for all sufficiently large r, as long as $\tau \in [16/\sqrt{r}, 1]$.*

Applying Fact 5.4.8, it follows that for the Mossell-O'Donnell function g_r on r inputs and any τ as above, we have

$$\frac{1}{2} \leq \mathrm{ExpBias}_{\gamma}(g_r) \leq \frac{1}{2} + \frac{1}{2}\sqrt{1 - K\tau} \leq 1 - \frac{K}{4}\tau, \tag{5.2}$$

for $\gamma \stackrel{\text{def}}{=} \frac{\tau}{\sqrt{r}}$.

We will use the above upper bound on expected bias together with the following key tool from [63], which gives a hardness amplification result for uniform distribution learning. This result builds on the original hardness amplification ideas of O'Donnell [146]. We note that the original theorem statement from [63] deals with the running time of learning algorithms, but inspection of the proof shows that the theorem also applies to the number of membership queries that the learning algorithms perform.

Theorem 5.4.10 (Theorem 12 of [63])**.** *Fix $g\colon \{0,1\}^r \to \{0,1\}$, and let $\mathcal{F}$ be a class of m-variable Boolean functions such that for every $f \in \mathcal{F}$, $\mathrm{bias}(f) \leq \frac{1}{2} + \frac{\epsilon}{8r}$. Let A be a uniform distribution membership query algorithm that learns $g \otimes \mathcal{F}$ to accuracy $\mathrm{ExpBias}_{\gamma}(g) + \epsilon$ using $T(m, r, 1/\epsilon, 1/\gamma)$ queries. Then there exists a uniform-distribution membership query algorithm B that learns $\mathcal{F}$ to accuracy $1 - \gamma$ using $O(T \cdot \mathrm{poly}(m, r, 1/\epsilon, 1/\gamma))$ membership queries.*

Hardness of learning monotone functions to high accuracy. At the bottom level, corresponding to step (a) in (5.1), our approach relies on the following simple claim which states that monotone functions are hard to learn to very high accuracy. (We view this claim as essentially folklore; as noted in the introduction, it slightly sharpens a lower bound given in [40]. A proof is given for completeness in Subsection 5.5.3.)

Claim 5.4.11 (A slice of hardness)**.** *There exists a class of balanced monotone Boolean functions $\mathcal{G} = \{\mathcal{G}_m\}_{m\in\mathbb{N}}$ and a universal constant C such that, for any constants $0 < \alpha \leq 1/10$, learning $\mathcal{G}_m$ to error $0 < \varepsilon \leq \alpha/\sqrt{m}$ requires at least 2^{Cm} membership queries.*

We now prove Theorem 5.4.1, i.e. we establish a stronger lower bound (in terms of the range of accuracy it applies to) against learning the class of *monotone* functions. We do this by amplifying the hardness result of Fact 5.4.11 by composing the "mildly hard" class of functions $\mathcal{G}$ with a monotone function g — the Mossel-O'Donnell function of Theorem 5.4.9 — that is very sensitive to small noise (intuitively, the noise rate here is comparable to the error rate from Fact 5.4.11).

Proof of Theorem 5.4.1. We will show that there exists an absolute constant $\alpha > 0$ such that for any n sufficiently large and $\tau \in [\frac{1}{n^{1/6}}, 1/2 - c]$, there exist $m = m(n)$, $r = r(n)$ (both of which are $\omega_n(1)$) such that learning the class of (balanced) functions $\mathcal{H}_n = g_r \otimes \mathcal{G}_m$ on $n = mr$ variables to accuracy $1 - \tau$ requires at least $2^{\alpha\sqrt{n}/\tau}$ membership queries.

By contradiction, suppose we have an algorithm A which, for all m, r, τ as above, learns the class $\mathcal{H}_n$ to accuracy $1 - \tau$ using $T = T_A(n, \tau) < 2^{\alpha\sqrt{n}/\tau}$ membership queries. We show that this implies that for infinitely many values of m, one can learn $\mathcal{G}_m$ to error $\varepsilon = .1/\sqrt{m}$ with $2^{o(m)}$ membership queries, in contradiction to Fact 5.4.11.

Fix any n large enough and $\tau \in [\frac{1}{n^{1/6}}, .1]$, and choose m, r satisfying $mr = n$ and $\frac{5}{K} \cdot \frac{\tau}{\sqrt{r}} = \frac{.1}{\sqrt{m}}$, where K is the constant from Theorem 5.4.9. Note that this implies $m = \frac{K}{50} \cdot \frac{\sqrt{n}}{\tau} \in [\Theta(n^{1/2}), \Theta(n^{2/3})]$ so indeed both m and r are $\omega_n(1)$. Intuitively, the value $\frac{.1}{\sqrt{m}}$ is the error we *want* to achieve to get a contradiction, while the value $\frac{5}{K} \cdot \frac{\tau}{\sqrt{r}}$ is the error we *can* get from Theorem 5.4.10. Note that we indeed can use the Mossel-O'Donnell function from Theorem 5.4.9, which requires $\tau > \frac{16}{\sqrt{r}}$ – for our choice of r, this is equivalent to $\tau > \left(\frac{16\sqrt{K}}{\sqrt{50}}\right)^{2/3} \frac{1}{n^{1/6}}$. Finally, set $\varepsilon \stackrel{\text{def}}{=} .1/\sqrt{m}$.

We apply Theorem 5.4.10 with $g \stackrel{\text{def}}{=} g_r$, $\gamma = (5/K)\tau/\sqrt{r}$ and $\epsilon = \tau/4$. Note that all functions in $\mathcal{G}_m$ are balanced, and thus trivially satisfy the condition that $\text{bias}(f) \leq \frac{\epsilon}{8r}$, and recall that $1-\gamma$ is the accuracy the theorem guarantees against the original class $\mathcal{G}_m$. With these parameters we have

$$\text{ExpBias}_\gamma(g) + \epsilon \underset{\text{Eq.(5.2)}}{\leq} 1 - \frac{K}{4}\frac{5\tau}{K} + \frac{\tau}{4} = 1 - \tau \leq \text{accuracy}(A).$$

Theorem 5.4.10 gives that there exists a learning algorithm B learning $\mathcal{G}_m$ to accuracy $1-\gamma \geq 1-\varepsilon$ with $T_B = O(T \cdot \text{poly}(m, r, 1/\tau, 1/\gamma)) = O(T \cdot \text{poly}(n, 1/\tau))$ membership queries, that is, $T_B = T_A(n,\tau) \cdot \text{poly}(n, 1/\tau) < 2^{\alpha\sqrt{n}/\tau + o(\sqrt{n}/\tau)}$ many queries. However, we have $2^{(\alpha+o(1))\sqrt{n}/\tau} = 2^{(\alpha+o(1))m \cdot \frac{\sqrt{n}}{\tau m}} < 2^{Cm}$, where the inequality comes from observing that $\frac{\sqrt{n}}{\tau m} = \frac{50}{K}$ (so that it suffices to pick α satisfying $50\alpha/K < C$). This contradicts Fact 5.4.11, and completes the proof of the theorem. □

Remark 3 (Improving this result)**.** *Proposition 1 of* [140] *gives a lower bound on the best noise stability that can be achieved by any monotone function. If this lower bound were in fact tight — that is, there exists a family of monotone functions $\{f_r\}$ such that for all $\gamma \in [-1,1]$, $\text{Stab}_{1-\gamma}(f_r) = (1-\gamma)^{(\sqrt{2/\pi}+o(1))\sqrt{r}}$ — then the above lower bound could be extended to an (almost) optimal range of τ, i.e. $\tau \in [\Phi(n)/\sqrt{n}, \frac{1}{2} - c]$ for Φ any fixed super-constant function.*

From the hardness of learning monotone functions to the hardness of learning k-alternating functions. We now establish the hardness of learning k-alternating functions. Hereafter we denote by $\mathcal{H} = \{g_r \otimes \mathcal{G}_m\}_{m,r}$ the class of "hard" monotone functions from Theorem 5.4.1. Since g_r is balanced and every $f \in \mathcal{G}_m$ has bias zero, it is easy to see that $\mathcal{H}$ is a class of balanced functions.

We begin by recalling the following useful fact about the noise stability of functions that are close to PAR:

Fact 5.4.12 (e.g., from the proof of Theorem 9 in [28])**.** *Let $r \geq 1$. If f is a Boolean function on r variables which η-approximates PAR_r, then for all $\delta \in [0,1]$,*

$$\text{Stab}_{1-2\delta}(f) \leq (1-2\eta)^2(1-2\delta)^r + 4\eta(1-\eta). \tag{5.3}$$

We use the above fact to define a function that is tailored to our needs: that is, a k-alternating function that is very sensitive to noise *and is defined on roughly k^2 inputs.* Without the last condition, one could just use PAR_k, but in our context this would only let us obtain a $\sqrt{k}$ (rather than a k) in the exponent of the lower bound, because of the loss in the reduction. To see why, observe that by using a combining function on k variables instead of k^2, the number of variables of the combined function $g_k \otimes \mathcal{G}_m$ would be only $n = km$. However, to get a contradiction with the hardness of monotone functions we shall need $k\sqrt{n}/\varepsilon \ll \sqrt{m}/\tau$, where $\tau \approx \varepsilon/k$, as the hardness amplification lemma requires the error to scale down with the number of combined functions.

Definition 5.4.13. *For any odd*[2] $r \geq k \geq 1$*, let* $\mathsf{PAR}'_{k,r}$ *be the symmetric Boolean function on r inputs defined as follows: for all $x \in \{0,1\}^r$,*

$$\mathsf{PAR}'_{k,r}(x) = \begin{cases} 0 & \text{if } |x| \leq \frac{r-k}{2} \\ 1 & \text{if } |x| \geq \frac{r+k}{2} \\ \mathsf{PAR}_r(x) & \text{otherwise.} \end{cases}$$

In particular, $\mathsf{PAR}'_{k,r}$ is k-alternating, and agrees with PAR_r on the $k+1$ middle layers of the hypercube. By an additive Chernoff bound, one can show that $\mathsf{PAR}'_{k,r}$ is η-close to PAR_r, for $\eta = e^{-k^2/2r}$.

Proof of Theorem 5.4.3. $\mathcal{H}_n^{(k)}$ will be defined as the class $\mathsf{PAR}'_{k,r} \otimes \mathcal{H}_m$ for some r and m such that $n = mr$ (see below). It is easy to check that functions in $\mathcal{H}_n^{(k)}$ are balanced and k-alternating. We show below that for n sufficiently large, $2 \leq k < n^{1/14}$ and $\varepsilon \in [(1/300)(k^{14}/n)^{1/6}, \frac{1}{2} - c]$, learning $\mathcal{H}_n^{(k)}$ to accuracy $1-\varepsilon$ requires $2^{\Omega\left(k\sqrt{n}/\varepsilon\right)}$ membership queries.

By contradiction, suppose we have an algorithm A learning for all n, k, ε as above the class of k-alternating functions to accuracy $1-\varepsilon$ using $T_A(n,k,\varepsilon) < 2^{\beta \frac{k\sqrt{n}}{\varepsilon}}$ membership

[2]The above definition can be straightforwardly extended to $r \geq k \geq 1$ not necessarily odd, resulting in a similar k-alternating perfectly balanced function $\mathsf{PAR}'_{k,r}$ that agrees with PAR_r on $k + O(1)$ middle layers of the cube and is 0 below and 1 above those layers. For the sake of simplicity we leave out the detailed description of the other cases.

queries, where $\beta > 0$ is a universal constant to be determined during the analysis. We claim that this implies that for infinitely many values of m, one can learn $\mathcal{H}_m$ to some range of accuracies with a number of membership queries contradicting the lower bound of Theorem 5.4.1.

Fix any n large enough, k and ε as above (which in particular impose $k = O(n^{1/14})$). The constraints we impose on m, r and τ are the following:

$$mr = n; \quad \mathrm{ExpBias}_\tau(\mathsf{PAR}'_{k,r}) + \varepsilon \leq 1 - \varepsilon; \quad m = \omega_n(1); \quad \tau \geq \frac{1}{m^{1/6}}; \tag{5.4}$$

$$\beta k \frac{\sqrt{n}}{\varepsilon} < \alpha \frac{\sqrt{m}}{\tau}, \tag{5.5}$$

where the constraints in (5.4) are for us to apply the previous theorems and lemmas, while (5.5) is needed to ultimately derive a contradiction.

One can show that by taking $r \stackrel{\text{def}}{=} \left\lfloor \frac{k^2}{2\ln 5} \right\rfloor \geq 1$ and $\tau \stackrel{\text{def}}{=} \frac{100\varepsilon}{r}$, the second constraint of (5.4) is satisfied, as then $\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r}) \leq 1 - 8\varepsilon$ (for the derivation, see Subsection 5.5.4). Then, with the first constraint of (5.4), we get (omitting for simplicity the floors) $m \stackrel{\text{def}}{=} \frac{n\tau}{100\varepsilon} = (2\ln 5)\frac{n}{k^2}$, so as long as $k = o(\sqrt{n})$, the third constraint of (5.4) is met as well. With these settings, the final constraint of (5.4) can be rewritten as $\varepsilon \geq \frac{1}{100}\left(\frac{r^7}{n}\right)^{1/6} = \frac{1}{100(2\ln 5)^{7/6}}\left(\frac{k^{14}}{n}\right)^{1/6}$. As $(2\ln 5)^{7/6} > 3$, it is sufficient to have $\varepsilon \geq \frac{1}{300}\left(\frac{k^{14}}{n}\right)^{1/6}$, which holds because of the lower bound on ε.

It only remains to check that Constraint (5.5) holds:

$$k\frac{\sqrt{n}}{\varepsilon} = 100k\frac{\sqrt{n}}{\tau r} = 100\frac{k}{\sqrt{r}}\frac{\sqrt{m}}{\tau} \leq \left(100\sqrt{\frac{2\ln 5}{1 - 2\ln 5/k^2}}\right)\frac{\sqrt{m}}{\tau} \leq 300\sqrt{2\ln 5}\cdot\frac{\sqrt{m}}{\tau},$$

where the first inequality holds because as $\frac{1}{r} \leq \frac{1}{\frac{k^2}{2\ln 5} - 1}$ and the second holds because $k \geq 2$. So for the right choice of $\beta = \Omega(1)$, e.g. $\beta = \alpha/600$, $\beta k\frac{\sqrt{n}}{\varepsilon} < \alpha\frac{\sqrt{m}}{\tau}$, and (5.5) is satisfied.

It now suffices to apply Theorem 5.4.10 to $\mathsf{PAR}'_{k,r} \otimes \mathcal{H}_m$, with parameters $\gamma = \tau$ and ε, on algorithm A, which has accuracy $\mathrm{acc}(A) \geq 1 - \tau \geq \mathrm{ExpBias}_\gamma(\mathsf{PAR}'_{k,r}) + \epsilon$. Since the functions of $\mathcal{H}$ are unbiased, it follows that there exists an algorithm B learning $\mathcal{H}_m$ to accuracy $1 - \tau$, with $\tau > 1/2m^{1/6}$, making only

$$T_B(m,\tau) = O(T_A(n,k,\varepsilon)\mathrm{poly}(n,k,1/\varepsilon)) = 2^{\beta k\frac{\sqrt{n}}{\varepsilon}(1+o(1))} < 2^{\alpha\frac{\sqrt{m}}{\tau}}$$

membership queries, which contradicts the lower bound of Theorem 5.4.1. □

Remark 4 (On the relation between ε and k)**.** *The tradeoff in the ranges for k and ε appear to be inherent to this approach. Namely, it comes essentially from Constraint (5.4), itself deriving from the hypotheses of Theorem 5.4.1. However, even getting an optimal range in the latter would still require $\tau = \Omega(1/\sqrt{m})$, which along with $r \approx k^2$ and $\tau \approx \varepsilon/r$ impose $k = O\big(n^{1/6}\big)$ and $\varepsilon = \Omega\big(k^3/\sqrt{n}\big)$.*

5.5 Auxiliary results

5.5.1 Proof of Claim 5.2.2

Suppose $\mathrm{Inf}[f] \geq \alpha n$ for some $\alpha \in (0,1]$: this means that at least an α fraction of all edges are bichromatic. Define the *weight level k* (denoted $\mathcal{W}_k$) to be the set of all edges going from a vertex of Hamming weight k to a vertex of Hamming weight $k+1$ (in particular, $|\mathcal{W}_k| = (n-k)\binom{n}{k}$), and consider weight levels $n/2 - a\sqrt{n}, \ldots, n/2 + a\sqrt{n}-1$ (the "middle levels") for $a \stackrel{\text{def}}{=} \sqrt{(1/2)\ln(8/\alpha)}$. (We suppose without loss of generality that $n/2 - a\sqrt{n}$ is a whole number.) Now, the fraction of all edges which do *not* lie in these middle levels is at most

$$\frac{1}{n2^{n-1}} \cdot 2 \sum_{j=0}^{\frac{n}{2}-a\sqrt{n}-1} |\mathcal{W}_k| \;\leq\; \frac{2n}{n2^{n-1}} \sum_{j=0}^{\frac{n}{2}-a\sqrt{n}-1} \binom{n}{k} \;\leq\; \frac{4}{2^n} \sum_{j=1}^{\frac{n}{2}-a\sqrt{n}-1} \binom{n}{k} \;\leq\; 4e^{-2a^2} = \frac{\alpha}{2}.$$

So no matter how many of these edges are bichromatic, it must still be the case that at least an $\alpha/2$ fraction of all edges in the "middle levels" are bichromatic.

Since the ratio

$$\frac{|\mathcal{W}_{n/2}|}{|\mathcal{W}_{n/2-a\sqrt{n}}|} \;=\; \frac{\frac{n}{2}\binom{n}{n/2}}{\left(\frac{n}{2}+a\sqrt{n}\right)\binom{n}{n/2-a\sqrt{n}}}$$

converges monotonically from below (when n goes to infinity) to $C \stackrel{\text{def}}{=} e^{2a^2}$, any two weight levels amongst the middle ones have roughly the same number of edges, up to a multiplicative factor C. Setting $p = \alpha/6C$ and $q = \alpha/6$, this implies that at least a p fraction of the weight levels in the middle levels have at least a q fraction of their edges being bichromatic. Indeed, otherwise we would have, letting b_k denote the number of bichromatic edges in

weight layer k,

$$\begin{aligned}
\frac{\alpha}{2} \cdot \underbrace{\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1} |\mathcal{W}_k|}_{\text{total}} &\leq \sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1} b_k \\
&\leq \sum_{\substack{k\in[\frac{n}{2}-a\sqrt{n},\frac{n}{2}+a\sqrt{n}-1] \\ b_k>q|\mathcal{W}_k|}} |\mathcal{W}_k| + \sum_{\substack{k\in[\frac{n}{2}a\sqrt{n},\frac{n}{2}+a\sqrt{n}-1] \\ b_k\leq q|\mathcal{W}_k|}} q\cdot|\mathcal{W}_k| \\
&\leq p\cdot 2a\sqrt{n}\cdot|\mathcal{W}_{n/2}| + q\cdot\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1}|\mathcal{W}_k| \\
&\leq p\cdot C\cdot\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1}|\mathcal{W}_k| + q\cdot\sum_{k=\frac{n}{2}-a\sqrt{n}}^{\frac{n}{2}+a\sqrt{n}-1}|\mathcal{W}_k|.
\end{aligned}$$

Thus $\frac{\alpha}{2}\cdot\text{total} \leq p\cdot C\cdot\text{total} + q\cdot\text{total}$, which gives $\frac{\alpha}{2} \leq \frac{\alpha}{6C}\cdot C + \frac{\alpha}{6} = \frac{\alpha}{3}$, a contradiction.

Let S be this collection of at least $2a\sqrt{n}p$ weight levels (from the middle ones) that each have at least a q fraction of edges being bichromatic, and write p_i to denote the fraction of bichromatic edges in $\mathcal{W}_i$, so that for each $i \in S$ it holds that $p_i \geq q$. Consider a random chain from 0^n to 1^n. The marginal distribution according to which an edge is drawn from any given fixed weight level i is uniform on $\mathcal{W}_i$, so by linearity, the expected number of bichromatic edges in a random chain is at least $\sum_{i\in S} p_i \geq 2a\sqrt{n}pq = \Omega(\sqrt{n})$, and hence some chain must have that many bichromatic edges. □

5.5.2 Derivation of Theorem 5.4.9 using Theorem 3 of [140]

The original theorem is stated for $\tau = 1$, with the upper bound being $1 - \Omega(1)$. However, the proof of [140] goes through for our purposes until the very end, where they set $\epsilon \stackrel{\text{def}}{=} \frac{1}{\sqrt{r}}$ and need to show that

$$e^{-2}\left(1 - (1 - \epsilon + 2\sqrt{\epsilon/r})^{\sqrt{r}}\right) = \Omega(1).$$

More precisely, the proof goes overall as follows: for some realization of the Talagrand function on r variables g_r, we want (for some absolute constant K) that

$$1 - K\tau \geq \text{Stab}_{1-\frac{\tau}{\sqrt{r}}}(g_r) = 1 - 2\Pr\left[g_r \circ N_{1-\frac{\tau}{\sqrt{r}}}(x) \neq g_r(x)\right].$$

That is, one needs to show $\Pr\Big[\, g_r \circ N_{1-\frac{\tau}{\sqrt{r}}}(x) \neq g_r(x) \,\Big] \geq \frac{K}{2}\tau$; and in turn, it is sufficient to prove that for g a random Talagrand function on r variables,

$$\mathrm{E}_g\Big[\, \Pr\Big[\, g \circ N_{1-\frac{\tau}{\sqrt{r}}}(x) \neq g(x) \,\Big] \,\Big] \geq \frac{K}{2}\tau.$$

This is where we slightly adapt the [140] proof. Where they set a parameter ϵ to be equal to $1/\sqrt{r}$ and analyze $\mathrm{E}_g[\Pr[\, g \circ N_{1-2\epsilon}(x) \neq g(x)\,]]$, we set for our purposes $\epsilon \stackrel{\text{def}}{=} \frac{\tau}{2\sqrt{r}}$. The rest of the argument goes through until the very end, where it only remains to show that

$$ae^{-2}\left(1 - (1 - \epsilon + 2\sqrt{\epsilon/r})^{\sqrt{r}}\right) \geq \frac{K}{2}\tau \tag{5.6}$$

(a being a small constant resulting from the various conditionings in their proof), or equivalently, that $(1 - \epsilon + 2\sqrt{\epsilon/r})^{\sqrt{r}} \leq 1 - \frac{e^2K}{2a}\tau$. But the left-hand side can be rewritten as

$$\begin{aligned}
(1 - \epsilon + 2\sqrt{\epsilon/r})^{\sqrt{r}} &= e^{\sqrt{r}\ln(1-\epsilon+2\sqrt{\epsilon/r})} = e^{\sqrt{r}\ln(1-\tau/2\sqrt{r}+\sqrt{2\tau}/r^{3/4})} \\
&= e^{\sqrt{r}\ln\left(1-\frac{\tau}{2\sqrt{r}}\left(1-\frac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right)\right)} \\
&\leq e^{-\sqrt{r}\cdot\frac{\tau}{2\sqrt{r}}\left(1-\frac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right)} && \left(\text{as } \tfrac{\tau}{2\sqrt{r}}\left(1-\tfrac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right) < 1\right) \\
&= e^{-\frac{\tau}{2}\left(1-\frac{2\sqrt{2}}{\sqrt{r^{1/2}\tau}}\right)} \leq e^{-\frac{\tau}{2}(1-\frac{1}{\sqrt{2}})} && \left(\text{as } \tau > \tfrac{16}{\sqrt{r}}\right) \\
&\leq e^{-\frac{\tau}{7}} \leq 1 - \frac{\tau}{8} \leq 1 - \frac{e^2K}{2a}\tau. \\
& && \text{(first as } \tau < 1\text{, then for a suitable choice of } K\text{)}
\end{aligned}$$

□

5.5.3 Proof of Fact 5.4.11

We give the proof for m even; by standard techniques, it extends easily to the odd case. For any $m \in 2\mathbb{N}$, define $\mathcal{C}_m$ as the class of functions f generated as follows: let $R = \{\, x \in \{0,1\}^m : \, |x| = m/2 \,\}$, and partition R in $|R|/2$ pairs of elements $(x^\ell, \bar{x}^\ell)$. For all $x \in \{0,1\}^m$,

$$f(x) = \begin{cases} 0 & \text{if } |x| < m/2 \\ r_\ell & \text{if } x \in R \text{ and } x = x^\ell \\ 1 - r_\ell & \text{if } x \in R \text{ and } x = \bar{x}^\ell \\ 1 & \text{if } |x| > m/2 \end{cases}$$

where the $|R|/2$ bits r_ℓ are chosen independently and uniformly at random. Clearly, f is balanced, and we have

$$|R| = \binom{m}{m/2} \underset{m\to\infty}{\sim} \sqrt{\frac{2}{\pi}} \cdot \frac{2^m}{\sqrt{m}} \stackrel{\text{def}}{=} \gamma 2^m.$$

Suppose we have a learning algorithm A for $\mathcal{C}_m$ making $q < 2^{Cm}$ membership queries. Fix $0 < \alpha \leq 1$, and $\varepsilon = \alpha/\sqrt{m}$; to achieve error at most ε overall, A must in particular achieve error at most $\frac{\varepsilon}{\gamma} = \sqrt{\frac{\pi}{2}}\alpha$ on R. But after making q queries, there are still at least $t = \gamma 2^m/2 - 2^{Cm} > 0.99|R|$ points in R (for m big enough) A has not queried, and hence with values chosen uniformly at random; on each of these points, A is wrong with probability exactly half, and in particular

$$\begin{aligned}
\Pr\left[\text{error } \leq \frac{\varepsilon}{\gamma}\right] &< \Pr[\text{error } \leq 2\alpha\,] \\
&= \Pr\left[\sum_{i=1}^{t} X_i \leq 2\alpha|R|\right] \\
&\leq \Pr\left[\sum_{i=1}^{t} X_i \leq \frac{200}{99}\alpha t\right] \\
&\leq e^{-\frac{(1-\frac{400}{99}\alpha)^2 t}{2}} \\
&= o(1),
\end{aligned}$$

with an additive Chernoff bound. This means that with high probability over the choice of the target concept, A will fail to learn it to accuracy $1-\varepsilon$. □

5.5.4 Derivation of the bound $\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r}) \leq 1 - 8\varepsilon$

By setting r as stated we get that $r \leq k^2/\ln(1/\varepsilon)$ and the distance between $\mathsf{PAR}'_{k,r}$ and PAR_r becomes $\eta = e^{-k^2/2r} \leq 1/5$. Since we aim at having $\mathrm{ExpBias}_\tau(\mathsf{PAR}'_{k,r}) \leq 1 - 2\varepsilon$, it is sufficient to have $\sqrt{\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r})} \leq 1 - 4\varepsilon$; which would in turn be implied by $\mathrm{Stab}_{1-\tau}(\mathsf{PAR}'_{k,r}) \leq 1 - 8\varepsilon$.

By Fact 5.4.12, it is sufficient to show that $(1-2\eta)^2(1-\tau)^r + 4\eta(1-\eta) \leq 1 - 8\varepsilon$. Note

that, since $\varepsilon < 1/100$, and by our choice of τ,

$$\begin{aligned}(1-2\eta)^2(1-\tau)^r + 4\eta(1-\eta) &\leq \frac{(1-2\eta)^2}{1+100\varepsilon} + 4\eta(1-\eta)\\ &\leq (1-2\eta)^2(1-50\varepsilon) + 4\eta(1-\eta)\\ &\leq (1-4\eta+4\eta^2)(1-50\varepsilon) + 4\eta(1-\eta)\\ &= 1-4\eta-50\varepsilon+200\eta\varepsilon+4\eta^2-200\varepsilon\eta^2+4\eta-4\eta^2\\ &= 1-50\varepsilon+200\varepsilon\eta(1-\eta) \leq 1-50\varepsilon+32\varepsilon = 1-18\varepsilon\\ &\leq 1-8\varepsilon.\end{aligned}$$

□

Chapter 6

The power of negations in Cryptography

6.1 Background, results, and organization

Why do block ciphers like AES (Advanced Encryption Standard) have so many XOR gates dispersed throughout the levels of its circuit? Can we build a universal hard-core bit alternative to the Goldreich and Levin one [80] that only applies a small (say, constant) number of XORs? Why does the Goldreich, Goldwasser, and Micali [82] construction of a pseudorandom function (PRF) from a pseudorandom generator (PRG) heavily rely on selection functions, and calls the PRG many times? Could there be a monotone construction of a PRF from a PRG?

These are a few of the many fascinating questions related to the negation complexity of cryptographic primitives. The *negation complexity* of a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is the minimum number of negation gates in any fan-in two circuit with AND, OR, and NOT gates computing f. Note that negation gates are equivalent to XOR gates (of fan-in 2), in the sense that any circuit with t negation gates can be transformed into an equivalent circuit with t XOR gates, and vice-versa.[1] A function is *monotone* if and only if its negation complexity is 0.

[1] $\neg x$ is equivalent to $x \oplus 1$, while $x \oplus y$ is equivalent to $\neg(x \wedge y) \wedge (x \vee y)$.

In this chapter, we initiate the investigation of the negation complexity of cryptographic primitives. We take first steps in this study, providing some surprising results, as well as pointing to some basic, intriguing problems that are still open.

This direction fits within the larger program of studying how *simple* basic cryptographic primitives can be, according to various complexity measures such as required assumptions, minimal circuit size, depth, etc (see, e.g., [18]). Exploring such questions helps gaining a deeper theoretical understanding of fundamental primitives and the relationships among them, and may provide the basis for understanding and addressing practical considerations as well.

While the study of monotone classes of functions and negation complexity has been prevalent in circuit complexity ([97, 15, 183, 181, 180, 26, 25, 139, 138], to name a few) and computational learning theory (see e.g. [29, 31, 40, 145, 54]), little attention has been given to it in the cryptographic context.

Recently, Goldreich and Izsak [79] have initiated a study of "cryptography in the monotone world", asking whether basic cryptographic primitives may be monotone. They focus on one-way functions (OWF) and pseudorandom generators, and show an inherent gap between the two by proving: (1) if any OWF exist, then there exist OWFs with polynomial-size monotone circuits, but (2) no monotone function can be a PRG. Quoting from their paper: *these two results indicate that in the "monotone world" there is a fundamental gap between one-way functions and pseudorandom generators; thus, the "hardness-vs-randomness" paradigm fails in the monotone setting.* This raises the following natural question:

> *Can other cryptographic primitives be computed by polynomial-size monotone circuits?*

We consider this question for several primitives and building blocks, showing negative answers for all of them. This may suggest the interpretation (or conjecture) that in the "monotone world", there is no cryptography except for one-way functions. We then initiate a *quantitative* study (where our main contributions lie), putting forward the question:

> *How many negations are required (for poly-size circuits) to compute fundamental cryptographic building blocks?*

Markov [132] proved that the negation complexity of any function $h\colon \{0,1\}^n \to \{0,1\}^m$ is at most $\lceil \log(n+1) \rceil$, and Fischer [66] proved that this transformation can be made efficient (see Jukna [Chapter 10, 109] for a modern exposition). In light of these results, is it the case that all natural cryptographic primitives other than OWFs require $\Omega(\log n)$ negations, or are there primitives that can be computed with, say, a constant number of negations?

We state our results informally below. Since our lower bounds hold for well-known primitives, we postpone their definitions to Section 6.2.

Our Results. Our contributions alongside previously known results are summarized in Figure 6.1, together with the main idea in each proof (the definition of these primitives can be found in Section 6.2). We explain and discuss some interesting aspects of these results below, deferring complete details to the body of the chapter.

Primitive	Lower Bound	Upper Bound	Ref.	Proof Techniques
OWF	-	(monotone)	[79]	Embedding into middle slice.
OWP	**non-monotone**	$\log n + O(1)$	-	Combinatorial and analytic proofs.
PRG	non-monotone	$\log n + O(1)$	[79]	AND of one or two output bits.
SBG	**non-monotone**	$\boldsymbol{\omega(1)}$	-	Extension of [79]; Parity of Tribes.
WPRF	non-monotone	$(\frac{1}{2}+o(1)) \log n$	[31]	Weak-learner for mon. functions.
PRF	$\boldsymbol{\log n - O(1)}$	$\log n + O(1)$	-	Alternating chains in the cube.
ECC	$\boldsymbol{\log n - O(1)}$	$\log n + O(1)$	-	Extension of [44].
HCB	$\boldsymbol{(\frac{1}{2}-o(1)) \log n}$	$(\frac{1}{2}+o(1)) \log n$	-	Low influence and [78].
EXT	$\boldsymbol{\Omega(\log n)}$	$\log n + O(1)$	-	Low noise-sensitivity and [36].

Figure 6.1: Summary of the negation complexity of basic cryptographic primitives and building blocks. Boldface results correspond to new bounds obtained in this work. The $\log n + O(1)$ upper bound is Markov's bound [132] for any Boolean function. Error-correcting codes (ECC) and extractors (EXT) refer to constructions with appropriate distance and extraction parameters.

Cryptography is Non-Monotone. As mentioned above, [79] proved that if OWFs exist, then they can be monotone, while PRGs cannot. We fill in the picture by considering several other cryptographic primitives, and observing that none of them can be monotone

(see Figure 6.1).

A result of particular interest is the lower bound showing that one-way permutations (OWP) *cannot* be monotone. We obtain this result by proving that any monotone permutation f on n variables must satisfy $f(x_1,\ldots,x_n) = \left(x_{\pi(1)},\ldots,x_{\pi(n)}\right)$, for some permutation $\pi\colon [n] \to [n]$ (finding π and inverting f can then be done by evaluating f on n inputs).[2] This is surprising in light of the [79] construction for OWFs. In particular, our result can be seen as a separation between OWFs and OWPs in the monotone world.

We provide two proofs of our result. The first is based on analytical methods, and was inspired by an approach used by Goldreich and Izsak [79]. The second is more elementary, and relies on a self-contained combinatorial argument.

Highly Non-Monotone Primitives. We show that many central cryptographic primitives are highly non-monotone. Some of our lower bounds demonstrate necessity of $\log n - O(1)$ negations, which is tight in light of Markov's $\log n + O(1)$ upper bound [132]. For some of the primitives we give less tight $\Omega(\log n)$ lower bounds.

Pseudorandom Functions (PRF). We show that PRFs can only be computed by circuits containing at least $\log n - O(1)$ negations (which is optimal up to the additive term). We prove this by exhibiting an adversary that distinguishes any function that can be implemented with fewer negations gates from a random function. Our result actually implies that for any PRF family $\{F(w,\cdot)\}$, for almost all seeds w, $F(w,\cdot)$ can only be computed by circuits with at least $\log n - O(1)$ negations.[3]

The distinguisher we construct asks for the values of the function on a fixed chain from 0^n to 1^n and accept if the alternating number of this chain is large. We note that the distinguisher suceeds for any function that has an implementation with fewer negations than the lower bound, regardless of the specific implementation the PRF designer had in mind. This can be considered as another statistical test to run on

[2] In order to avoid confusion, observe that by assumption f permutes n-bit strings, while π is permuting the indexes of the input variables.

[3] That is, if we consider the circuit computing the PRF family $F(\cdot,\cdot)$ as a single function (with the seed as one of the inputs), then this circuit must have at least logarithmically many negation gates.

proposed candidate PRF implementations.

Error-Correcting Codes (ECC). As shown by Buresh-Oppenheim, Kabanets and Santhanam [44], if an ECC has a monotone encoding function then one can find two codewords that are very close. This implies that there is no monotone ECC with good distance parameters.

We extend this result to show that, given a circuit with t negation gates computing the encoding function, we can find two codewords whose Hamming distance is $O(2^t \cdot m/n)$ (for codes going from n bits to m bits). Consequently, this gives a $\log n - O(1)$ lower bound on the negation complexity of ECC with optimal distance parameters.

Hard-core Bits (HCB). Recall that a Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ is a hard-core predicate for a function $f\colon \{0,1\}^n \to \{0,1\}$ if, given $f(x)$, it is hard to compute $h(x)$. We show that general hard-core bit predicates must be highly non-monotone. More specifically, there exists a family of one-way functions f_n for which any hard-core predicate requires $\Omega(\log n)$ negations (assuming one-way functions exist).

Our result follows via the analysis of the *influence* of circuits with few negations, and a corresponding lower bound on hard-core bits due to Goldmann and Russell [78].

(Strong) Extractors (EXT). A strong extractor produces almost uniform bits from weak sources of randomness, even when the truly random seed used for extraction is revealed. We prove that any extractor function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^{100}$ that works for $(n, n^{1/2-\varepsilon})$-sources requires circuits with $\Omega(\log n)$ negations (see Section 6.2 for definitions).

This proof relies on the analysis of the *noise sensitivity* of circuits containing negations, together with a technique from Bogdanov and Guo [36].

Non-Trivial Upper Bound for Small-Bias Generators. The above lower bounds may suggest the possibility that, with the exception of OWFs, all cryptographic building blocks require $\Omega(\log n)$ negations. We show one example of a primitive – small-bias generator (SBG) – that can be constructed with significantly fewer negations, namely, with any super-constant number of negations (for example, $\log^*(n)$ such gates).

A SBG can be thought of as a weaker version of a PRG, where the output fools linear distinguishers (i.e., it looks random to any distinguisher that can only apply a linear test). Thus, any PRG is also a SBG, but not vice-versa. We construct our SBG with few negations by outputting the input and an additional bit consisting of a parity of independent copies of the Tribes function.

Since SBGs are not quite a cryptographic primitive (these can be constructed unconditionally, and are not secure against polynomial adversaries), one may still conjecture that all "true" cryptographic primitives with the exception of OWFs require $\Omega(\log n)$ negations. We do not know whether this is the case, and it would be interesting to determine whether other primitives not covered in this chapter can be monotone.

Lower Bounds for Boolean Circuits with Bottom Negations. In addition to studying specific primitives, we investigate general structural properties of circuits with negations. We prove a theorem showing that for monotone functions, the depth of any circuit with negations at the bottom (input) level only is lower bounded by the monotone depth complexity of the function minus the number of negations in the circuit. This is connected to a result obtained by Koroth and Sarma [124], who proved a multiplicative rather than additive lower bound, but in a more general setting which assumes that every Boolean function computed at an internal gate of the circuit can be computed by some circuit with few negations (see their paper for more details). We consider the usual definition of Boolean circuits with negations at the bottom layer, which allows us to prove a stronger trade-off. Our proof is inspired by ideas from [124], and relies on a circular application of the Karchmer-Wigderson connection between boolean circuits and communication protocols.

This result suggests that negations at the bottom layer are less powerful and easier to study. In Section 6.6 we describe some techniques (following results of Blais et al. [29]) that allow one to decompose arbitrary computations into monotone and non-monotone components, and provide further evidence that negations at the bottom are less powerful (see also the discussion in Section 6.5).

Organization of the Chapter. We provide the definitions for most of the primitives mentioned in this chapter in Section 6.2. Basic results used later in our proofs are presented

in Section 6.3, with some proofs deferred to Section 6.6. Our main results appear in Section 6.4. Finally, Section 6.5 discusses some open problems motivated by our work.

6.2 Preliminaries and notation

In this section, we set up notation and define relevant concepts. We refer the reader to the textbooks [109], [19], [83], and [128] for more background in circuit complexity, computational complexity, theory of cryptography, and communication complexity, respectively.

6.2.1 Basic notation

Unless explicitly stated, we assume that the underlying probability distribution in our equations is the uniform distribution over the appropriate set. Further, we let $\mathcal{U}_\ell$ denote the uniform distribution over $\{0,1\}^\ell$. We use $\log x$ to denote a logarithm in base 2, and $\ln x$ to refer to the natural base.

For convenience of the reader, we review of a few relevant definitions from Chapter 5. Given strings $x, y \in \{0,1\}^n$, we write $x \preceq y$ if $x_i \leq y_i$ for every $i \in [n]$. A *chain* $\mathcal{X} = (x^1, \ldots, x^t)$ is a monotone sequence of strings over $\{0,1\}^n$, i.e., $x^i \preceq x^{i+1}$ for every $i \in [1, t-1]$. We say that a chain $\mathcal{X} = (x^1, x^2, \ldots, x^t)$ is *k-alternating* with respect to a function $f\colon \{0,1\}^n \to \{0,1\}$ if there exist indexes $i_0 < i_1 < \ldots < i_k$ such that $f(x^{i_j}) \neq f(x^{i_j+1})$, for every $j \in [0, k-1]$. If this is true for every pair of consecutive elements of the chain, we say that the chain is *proper* (with respect to f). We let $a(f, \mathcal{X})$ be the size of the largest set of indexes satisfying this condition. The alternating complexity of a Boolean function f is given by $a(f) \stackrel{\text{def}}{=} \max_{\mathcal{X}} a(f, \mathcal{X})$, where $\mathcal{X}$ is a chain over $\{0,1\}^n$. A function $f\colon \{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \leq f(y)$ whenever $x \preceq y$. A function $g\colon \{0,1\}^n \to \{0,1\}^m$ is monotone if every output bit of g is a monotone Boolean function. Moreover, we say that a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is *anti-monotone* if f is the negation of a monotone Boolean function.

6.2.2 Boolean circuits and negation gates

Every Boolean circuit mentioned in this chapter consists of AND, OR and NOT gates, where the first two types of gates have fan-in two. Recall that a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone if and only if it is computed by a circuit with AND and OR gates only.

For convenience, the size of a circuit C will be measured by its number of AND and OR gates, and will be denoted by $\mathsf{size}(C)$. The depth of a circuit C, denoted by $\mathsf{depth}(C)$, is the largest number of AND and OR gates in any path from the output gate to an input variable. The depth of a Boolean function f is the minimum depth of a Boolean circuit computing f. Similarly, the depth of a *monotone* function f, denoted by $\mathsf{depth}^+(f)$, is the minimum depth among all *monotone* circuits computing f. We will also consider multi-output Boolean circuits that compute Boolean functions $f\colon \{0,1\}^n \to \{0,1\}^m$. We stress that whenever we say that a function of this form is computed by a circuit with t negations, it means that there exists a *single* circuit (with multiple output gates) containing at most t negations computing f.

We say that a circuit contains negation gates at the bottom layer only if any NOT gate in the circuit gets as input an input variable x_i, for some $i \in [n]$. We will also say that circuits of this form are DeMorgan circuits. Put another way, a circuit $C(x)$ of size s with t negations at the bottom layer can be written as $D(x, (x \oplus \beta))$, where D is a *monotone* circuit of size s, $\beta \in \{0,1\}^n$ with $|\beta|_1 = t$ encodes the variables that appear negated in C, and $x \oplus \beta \in \{0,1\}^n$ is the string obtained via the bit-wise XOR operation. This notation is borrowed from Koroth and Sarma [124], which refers to β as the orientation vector.

6.2.3 Pseudorandom functions and weak pseudorandom functions

Let $\mathcal{F}_n$ be the set of all Boolean functions on n variables, and $F\colon \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$. We say that F is an (s,ε)-secure *pseudorandom function* (PRF) if, for every (non-uniform) algorithm $\mathcal{A}$ that can be implemented by a circuit of size at most s,

$$\left| \Pr_{w \sim \{0,1\}^m} \left[\mathcal{A}^{F(w,\cdot)} = 1 \right] - \Pr_{f \sim \mathcal{F}_n} \left[\mathcal{A}^{f} = 1 \right] \right| \leq \varepsilon,$$

where $\mathcal{A}^h$ denotes the execution of $\mathcal{A}$ with oracle access to a Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ (circuits with access to oracle gates are defined in the natural way).

A *weak pseudorandom function* (WPRF) is defined similarly, except that the distinguisher only has access to *random examples* of the form $(x, F(w,x))$, where x is uniformly distributed over $\{0,1\}^n$. In particular, any (s,ε)-secure pseudorandom function is an (s,ε)-secure weak pseudorandom function, while the other direction is not necessarily true.

6.2.4 Pseudorandom generators and small-bias generators

A function $G\colon \{0,1\}^n \to \{0,1\}^m$ is an (s,ε)-secure *pseudorandom generator* (PRG) with *stretch* $\ell \stackrel{\text{def}}{=} m-n$ if for every circuit $C(z_1,\ldots,z_m)$ of size s,

$$\left| \Pr_{x\sim\mathcal{U}_n}[C(G(x))=1] - \Pr_{y\sim\mathcal{U}_m}[C(y)=1] \right| \leq \varepsilon.$$

We say that a function $g\colon \{0,1\}^n \to \{0,1\}^m$ is an ε-secure *small-bias generator* (SBG) with stretch $\ell = m-n$ if, for every nonempty set $S \subseteq [m]$,

$$\left| \Pr_{x\sim\mathcal{U}_n,\, y=g(x)} \Big[\sum_{i\in S} y_i \equiv 1 \pmod 2 \Big] - \frac{1}{2} \right| \leq \varepsilon.$$

Observe that small-bias generators can be seen as weaker pseudorandom generators that are required to be secure against linear distinguishers only. We refer the reader to Naor and Naor [141] for more information about the constructions and applications of such generators.

6.2.5 One-way functions, one-way permutations, and hard-core bits

We say that a function $f\colon \{0,1\}^n \to \{0,1\}^m$ is an (s,ε)-secure *one-way function* (OWF) if for every circuit C of size at most s,

$$\Pr_{x\sim\mathcal{U}_n,\, y=f(x)}[C(y) \in f^{-1}(y)] \leq \varepsilon.$$

If $m=n$, we say that f is *length-preserving*. If in addition f is a one-to-one mapping, we say that f is an (s,ε)-secure *one-way permutation* (OWP).

We say that a function $h\colon \{0,1\}^n \to \{0,1\}$ is an (s,ε)-secure hard-core bit for a function $f\colon \{0,1\}^n \to \{0,1\}^m$ if, for every circuit C of size s,

$$\left| \Pr_{x\sim\mathcal{U}_n}[C(f(x))=h(x)] - \frac{1}{2} \right| \leq \varepsilon.$$

6.2.6 Extractors and error-correcting codes

The *min-entropy* of a random variable X, denoted by $H_\infty(X)$, is the largest real number k such that $\Pr[X = x] \leq 2^{-k}$ for every x in the range of X. A distribution X over $\{0,1\}^n$ with $H_\infty(X) \geq k$ is said to be an (n,k)-source. Given random variables X and Y with range $\{0,1\}^m$, we let

$$\delta(X,Y) \stackrel{\text{def}}{=} \max_{S \subseteq \{0,1\}^m} \left| \Pr[X \in S] - \Pr[Y \in S] \right|$$

denote their *statistical distance*. We say that X and Y are ε-close if $\delta(X,Y) \leq \varepsilon$.

We say that a function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ is a (strong) (k,ε)-*extractor* (EXT) if, for any (n,k)-source X, the distributions $\mathcal{U}_{s+m}$ and $(\mathcal{U}_s, \mathsf{Ext}(X,\mathcal{U}_s))$ are ε-close.[4]

Given strings $y^1, y^2 \in \{0,1\}^m$, we let

$$\Delta(y^1,y^2) \stackrel{\text{def}}{=} \frac{|\{i \in [m] \mid y^1_i \neq y^2_i\}|}{m}$$

be their *relative Hamming distance*. Given a function $E\colon \{0,1\}^n \to \{0,1\}^m$, we say that E has *relative distance* γ if for every distinct pair of inputs $x^1, x^2 \in \{0,1\}^n$, we have $\Delta(E(x^1), E(x^2)) \geq \gamma$. As a convention, we will refer to a function of this form as an *error-correcting code* (ECC) whenever we are interested in the distance between its output strings (also known as "codewords").

6.3 Basic results and technical background

6.3.1 Markov's upper bound

The following result was obtained by Markov [132].

Proposition 6.3.1 (Markov [132])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}^m$ be an arbitrary function. Then f is computed by a (multi-output) Boolean circuit containing at most $\lceil \log(n+1) \rceil$ negations.*

This result implies that many of our lower bounds are tight up to an additive term independent of n. Some of our proofs also rely on the following relation between negation complexity and alternation.

[4] Two occurrences of the same random variable in an expression refer to the same copy of the variable.

Proposition 6.3.2 (Markov [132])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit with at most t negations. Then $a(f) = O(2^t)$.*

6.3.2 The flow of negation gates

It is useful in some situations to decompose the computation of a function into monotone and non-monotone components. This idea has been applied successfully in Chapter 5 to obtain almost optimal bounds on the learnability of functions computed with few negation gates. A useful structural result employed there (Corollary 5.1.2), restated below as a lemma for convenience of the reader, is the following.

Lemma 6.3.3 (Blais et al. [29])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C with at most t negations. Then f can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is monotone, $T = O(2^t)$, and h is either the parity function, or its negation.*

A drawback of this statement is that the computational complexity of each g_i is not related to the size of C. Roughly speaking, the proof of this result uses a circuit for f in order to gain structural information about f, and then rely on a non-constructive argument. We observe that, by relaxing the assumption on h, we can prove the following effective version of Lemma 6.3.3.[5]

Lemma 6.3.4. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C of size s containing t negation gates. Then f can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is computed by a monotone circuit of size at most s, $T = 2^{t+1} - 1$, and $h\colon \{0,1\}^T \to \{0,1\}$ is computed by a circuit of size at most $5T$.*

We state below a more explicit version of Lemma 6.3.4 for circuits with a single negation gate and several output bits. The proof of this result follows from the same argument used to derive Lemma 6.3.4.

Lemma 6.3.5. *Let $f\colon \{0,1\}^n \to \{0,1\}^u$ be computed by a circuit of size s containing a single negation gate. Assume that the j-th output bit of f is computed by the function $f_j\colon \{0,1\}^n \to \{0,1\}$. Then, there exist monotone functions $m\colon \{0,1\}^n \to \{0,1\}$ and*

[5] This result was obtained during a discussion with Clement Canonne, Li-Yang Tan, and Rocco Servedio.

$m_{j,\ell}\colon \{0,1\}^n \to \{0,1\}$, *where* $j \in [u]$ *and* $\ell \in \{0,1\}$, *which are computed by monotone circuits of size at most* s, *and a function* $h\colon \{0,1\}^3 \to \{0,1\}$, *such that:*

(*i*) *For every* $j \in [u]$, $f_j(x) = h(m(x), m_{j,0}(x), m_{j,1}(x))$.

(*ii*) *For every* $j \in [u]$ *and* $x \in \{0,1\}^n$, $m_{j,0}(x) \leq m_{j,1}(x)$.

(*iii*) *The function* h *is defined as* $h(z, y_1, y_0) \stackrel{\text{def}}{=} y_z$.

From a programming perspective, Lemma 6.3.5 shows that a single negation gate in a Boolean circuit can be interpreted as an IF-THEN-ELSE statement involving monotone functions. Conversely, the selection procedure computed by h can be implemented with a single negation.

For convenience of the reader, we sketch the proof of these results in Section 6.6, where we also discuss the expressiveness of negations at arbitrary locations compared to negations at the bottom layer of a circuit. Lemmas 6.3.3 and 6.3.4 can be used interchangeably in our proofs.

6.3.3 Useful inequalities

Some of our proofs rely on the following results for Boolean functions.

Proposition 6.3.6 (Fortuin, Kasteleyn, and Ginibre [73])**.** *If* $g\colon \{0,1\}^n \to \{0,1\}$ *and* $f\colon \{0,1\}^n \to \{0,1\}$ *are monotone Boolean functions, then*

$$\Pr_x[f(x) = 1 \wedge g(x) = 1] \geq \Pr_x[f(x) = 1] \cdot \Pr_x[g(x) = 1].$$

The same inequality holds for anti-monotone functions. In particular, for monotone functions $f, g\colon \{0,1\}^n \to \{0,1\}$, *following inequality holds:*

$$\Pr_x[f(x) = 0 \wedge g(x) = 0] \geq \Pr_x[f(x) = 0] \cdot \Pr_x[g(x) = 0].$$

A stronger version of this inequality that will be used in some of our proofs is presented below.

Proposition 6.3.7 (Talagrand [182])**.** *For any pair of monotone Boolean functions $f, g\colon \{0,1\}^n \to \{0,1\}$, it holds that*

$$\Pr_x[f(x)=1 \wedge g(x)=1] \geq \Pr_x[f(x)=1]\cdot\Pr_x[g(x)=1] + \psi\Big(\sum_{i\in[n]} \mathrm{Inf}_i(f)\cdot\mathrm{Inf}_i(g)\Big),$$

where $\psi(x) \stackrel{\text{def}}{=} c\cdot x/\log(e/x)$, and $c>0$ is a fixed constant independent of n.

Proposition 6.3.8 (Kahn, Kalai, and Linial [113])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a balanced Boolean function. Then there exists an index $i\in[n]$ such that $\mathrm{Inf}_i(f) = \Omega\Big(\frac{\log n}{n}\Big)$.*

6.4 Lower bounds on negation complexity

6.4.1 One-way functions versus one-way permutations

Goldreich and Izsak [79] proved that if one-way functions exist, then there are *monotone* one-way functions. We show below that this is not true for *one-way permutations*. In other words, one-way permutations are inherently non-monotone. This lower bound follows easily via the following structural result for monotone permutations.

Proposition 6.4.1. *Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be a one-to-one function. If f is monotone, then there exists a permutation $\pi\colon [n]\to[n]$ such that, for every $x\in\{0,1\}^n$, $f(x) = x_{\pi(1)}\ldots x_{\pi(n)}$. In particular, there exists a (uniform) polynomial size circuit that inverts f on every input $y=f(x)$.*

Proof. Let $f_i\colon \{0,1\}^n \to \{0,1\}$ be the Boolean function corresponding to the i-th output bit of f. Since f is monotone, each function f_i is monotone. Consider now functions f_ℓ and f_k, where $\ell \neq k$. By Talagrand's inequality (Proposition 6.3.7),

$$\Pr_x[f_\ell(x)=1\wedge f_k(x)=1] \geq \Pr_x[f_\ell(x)=1]\cdot\Pr_x[f_k(x)=1] + \psi\Big(\sum_{i\in[n]}\mathrm{Inf}_i(f_\ell)\cdot\mathrm{Inf}_i(f_k)\Big). \quad (6.1)$$

Since f is a permutation, $\Pr_x[f_\ell(x)=1\wedge f_k(x)=1] = 1/4$ and $\Pr_x[f_\ell(x)=1] = \Pr_x[f_k(x)=1] = 1/2$. Consequently, it follows from Equation 6.1 and the definition of ψ that

$$\sum_{i\in[n]}\mathrm{Inf}_i(f_\ell)\cdot\mathrm{Inf}_i(f_k) = 0.$$

In other words, f_ℓ and f_k depend on a disjoint set of input variables. Since this is true for every pair ℓ and k with $\ell \neq k$, and every output bit of f is non-constant, there exists a permutation $\pi\colon [n] \to [n]$ such that, for every $i, j \in [n]$, if $\mathrm{Inf}_i(f_j) > 0$ then $i = \pi(j)$. Moreover, as f is monotone and one-to-one, we must have $f_j(x) = x_{\pi(j)}$, for every $j \in [n]$. The corresponding permutation can be easily recovered from f by evaluating this function on every indicator string $e^i \in \{0,1\}^n$, where $e^i_j = 1$ if and only if $i = j$. This completes the proof of our result. □

We remark that a simple extension of this proof allows us to rule out monotone one-way functions $f\colon \{0,1\}^n \to \{0,1\}^{n-k}$ where each pre-image set $f^{-1}(x)$ has size exactly 2^k (i.e., regular OWFs).

Proposition 6.4.1 implies that any circuit computing a one-way permutation contains at least one negation gate. It is not clear how to extend its proof to obtain a stronger lower bound on the negation complexity of one-way permutations, as Talagrand's inequality holds for monotone functions only. Although we leave open the problem of obtaining better lower bounds, we give next an alternative proof of Proposition 6.4.1 that does not rely on Talagrand's result.

Proof. Let $S_k \stackrel{\text{def}}{=} \{x \in \{0,1\}^n \mid |x|_1 = k\}$, where $k \in [0, n]$. In other words, S_k is simply the k-th slice of the n-dimensional Boolean hypercube. Initially, we prove the following claim: For every set S_k, $f(S_k) = S_k$. In other words, f induces a permutation over each set of inputs S_k. We then use this result to establish Proposition 6.4.1.

First, observe that $f(0^n) = 0^n$. Otherwise, there exists an input $x \neq 0^n$ such that $f(x) = 0^n$. Since $0^n \preceq x$ and f is monotone, we get that $f(0^n) \preceq f(x)$, which contradicts the injectivity of f. This establishes the claim for S_0. The general case follows by induction on k. Assume the result holds for any $k' < k$, and consider an arbitrary $y \in S_k$. Since f is one-to-one, there exists $x \in S_\ell$ such that $f(x) = y$, where $\ell \geq k$. If $\ell \neq k$, there exists $x' \prec x$ such that $x' \in S_k$. Let $y' \stackrel{\text{def}}{=} f(x')$. Using that f is monotone and $x' \prec x$, we get that $y' \preceq y$. Since f is one-to-one, $y' \prec y$, thus $y' \in S_{k'}$ for some $k' < k$. This is in contradiction with our induction hypothesis and the injectivity of f, since $f(S_{k'}) = S_{k'}$, $x' \in S_k$, and $y' = f(x') \in S_{k'}$. This completes the induction hypothesis, and the proof of our claim.

Now let $\pi\colon [n] \to [n]$ be the permutation such that $f^{-1}(e^i) = e^{\pi(i)}$, where $e^j \in \{0,1\}^n$ is the input with 1 at the j-th coordinate only. Clearly, for every $x \in S_0 \cup S_1$, $f(x) = x_{\pi(1)} \ldots, x_{\pi(n)}$. On the other hand, for every $x \in S_k$ with $k > 1$, it follows from the monotonicity of f that

$$\bigvee_{i\,:\,x_i=1} f(e^i) \;\preceq\; f(x),$$

where the disjunction is done coordinate-wise. Finally, it follows from our previous claim that we must also have $f(x) \in S_{|x|_1}$. Therefore,

$$\bigvee_{i\,:\,x_i=1} f(e^i) \;=\; f(x).$$

Consequently, for every $x \in \{0,1\}^n$, it follows that $f(x) = x_{\pi(1)} \ldots x_{\pi(n)}$, which completes the proof. □

6.4.2 Pseudorandom generators and small-bias generators

In contrast to the situation for one-way functions, Goldreich and Izsak [79] presented an elegant proof that pseudorandom generators cannot be monotone. More specifically, their result shows that the output distribution of a monotone function $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$ can be distinguished from random either by the projection of one of its output bits, or via the conjunction of two output bits.

Recall from Section 6.2 that small-bias generators can be seen as restricted pseudorandom generators that are only required to be secure against linear tests. We prove next that the techniques from [79] can be used to show that there are no $(1/n^{\omega(1)})$-secure monotone small-bias generators with 1 bit of stretch. We observe later in this section that such generators can be constructed with any super-constant number of negation gates.

Proposition 6.4.2. *For any monotone function $G\colon \{0,1\}^n \to \{0,1\}^{n+1}$, there exists a (non-uniform) linear test $D\colon \{0,1\}^{n+1} \to \{0,1\}$ such that*

$$\left| \Pr_{x \sim \mathcal{U}_n} [D(G(x)) = 1] - \frac{1}{2} \right| = \Omega\left(\frac{1}{n^2}\right).$$

Proof. The proof follows closely the argument in [79], combined with an appropriate application of the FKG inequality (Proposition 6.3.6). Let $G_i\colon \{0,1\}^n \to \{0,1\}$ be the Boolean

function corresponding to the i-th output bit of G, where $i \in [n+1]$. Observe that if there exists i such that

$$\left| \Pr_{x \sim \mathcal{U}_n}[G_i(x) = 1] - \frac{1}{2} \right| = \Omega\left(\frac{1}{n^2}\right),$$

then there is a trivial linear distinguisher for G.

Assume therefore that, for every $i \in [n+1]$, G_i is almost balanced. In particular, each function G_i is $\delta(n)$-close under the uniform distribution to an unbiased function $\widetilde{G}_i \colon \{0,1\}^{n+1} \to \{0,1\}$, where $\delta(n) = o((\log n)/n)$. It follows from Proposition 6.3.8 that each function $\widetilde{G}_i$ has an influential variable. More precisely, there exists $\gamma \colon [n+1] \to [n]$ such that

$$\mathrm{Inf}_{\gamma(i)}(\widetilde{G}_i) = \Omega\left(\frac{\log n}{n}\right),$$

for every $i \in [n+1]$. As each G_i is $\delta(n)$-close to $\widetilde{G}_i$, it follows that $\mathrm{Inf}_{\gamma(i)}(G_i) = \Omega\left(\frac{\log n}{n}\right)$ as well.

By the pigeonhole principle, there exist distinct indexes i and j such that $\gamma(i) = \gamma(j)$. It follows from Proposition 6.3.7 that

$$\begin{aligned}
\Pr_x[G_i(x) = 1 \wedge G_j(x) = 1] &\geq \Pr_x[G_i(x) = 1] \cdot \Pr_x[G_j(x) = 1] + \psi\Big(\sum_{k \in [n]} \mathrm{Inf}_k(G_i) \cdot \mathrm{Inf}_k(G_k)\Big) \\
&\geq \Pr_x[G_i(x) = 1] \cdot \Pr_x[G_j(x) = 1] + \Omega\Big(\psi\Big(\frac{\log^2 n}{n^2}\Big)\Big) \\
&\geq \Pr_x[G_i(x) = 1] \cdot \Pr_x[G_j(x) = 1] + \Omega\Big(\frac{\log n}{n^2}\Big).
\end{aligned}$$

On the other hand, Proposition 6.3.6 implies that

$$\Pr_x[G_i(x) = 0 \wedge G_j(x) = 0] \geq \Pr_x[G_i(x) = 0] \cdot \Pr_x[G_j(x) = 0].$$

Combining both inequalities, and using the assumption that each output bit of G is almost

balanced, we get that:

$$\begin{aligned}
\Pr_x[G_i(x)+G_j(x)=0] &= \Pr_x[G_i(x)=1 \wedge G_j(x)=1] + \Pr_x[G_i(x)=0 \wedge G_j(x)=0] \\
&\geq \Pr_x[G_i(x)=1] \cdot \Pr_x[G_j(x)=1] \\
&+ \Pr_x[G_i(x)=0] \cdot \Pr_x[G_j(x)=0] + \Omega\Big(\frac{\log n}{n^2}\Big) \\
&\geq \frac{1}{2} - O\left(\frac{1}{n^2}\right) + \Omega\Big(\frac{\log n}{n^2}\Big) \\
&= \frac{1}{2} + \Omega\Big(\frac{\log n}{n^2}\Big).
\end{aligned}$$

Therefore, the linear function $D(y) \stackrel{\text{def}}{=} y_i + y_j$ can distinguish the output of G from random with the desired advantage, which completes the proof. □

In contrast, we show next that there *are* small-bias generators with super-polynomial security that can be computed with *any* super-constant number of negations. Let

$$\mathsf{Tribes}_{s,t} \colon \{0,1\}^{s \cdot t} \to \{0,1\}$$

be the (monotone) Boolean function defined by

$$\mathsf{Tribes}_{s,t}(x_1, \ldots, x_{s \cdot t}) = \bigvee_{i=0}^{s-1} \bigwedge_{j=1}^{t} x_{i \cdot t + j}.$$

Further, we use $\mathsf{Tribes}_m \colon \{0,1\}^m \to \{0,1\}$ to denote the function $\mathsf{Tribes}_{s,t}$, where s is the largest integer such that $1-(1-2^{-t})^s \leq 1/2$, and $t = m/s$ (i.e., we try to make Tribes as balanced as possible as a function over m variables).

Proposition 6.4.3. *Let $f \colon \{0,1\}^n \to \{0,1\}$ be defined as $f(x) \stackrel{\text{def}}{=} \oplus_{i=1}^{k} \mathsf{Tribes}_{n/k}(x^{(i)})$, where $x^{(i)}$ denotes the i-th block of x with length n/k. Let $1 \leq k(n) \leq n/\log n$, and $G \colon \{0,1\}^n \to \{0,1\}^{n+1}$ be defined by $G(x) \stackrel{\text{def}}{=} (x, f(x))$, Then, there exists a constant $C > 0$ such that, for any linear function $D \colon \{0,1\}^{n+1} \to \{0,1\}$,*

$$\left| \Pr_{x \sim \mathcal{U}_n}[D(G(x)) = 1] - \frac{1}{2} \right| \leq \left(C \cdot (k/n) \cdot \log(n/k)\right)^k.$$

In particular, when $k = \omega(1)$, we can get a small-bias generator with negligible error that can be computed with roughly $\log k$ negations (via Proposition 6.3.1). Interestingly, for $k = 2$ we obtain an SBG computed with a *single negation* and security $\tilde{\Theta}(n^{-2})$, essentially matching the lower bound for *monotone* SBGs given by Proposition 6.4.2.

Proof. We assume the reader is familiar with basic concepts from analysis of Boolean functions (cf. O'Donnell [147]). Suppose that $D(y) \stackrel{\text{def}}{=} \sum_{i \in S} y_i \pmod 2$, where $S \subseteq [n+1]$ is nonempty. If $n+1 \notin S$, using that the first n output bits of G are uniformly distributed over $\{0,1\}^n$, we get that $|\Pr_x[D(G(x)) = 1] - 1/2| = 0$. Assume therefore that $n+1 \in S$, and let $S' \stackrel{\text{def}}{=} S \backslash \{n+1\}$. Then,

$$\begin{aligned}
\Big| \Pr_{x \sim \mathcal{U}_n} \Big[D(G(x)) = 1 \Big] - \frac{1}{2} \Big| &= \Big| \Pr_{x \sim \mathcal{U}_n} \Big[f(x) + \sum_{i \in S'} x_i \equiv 1 \pmod 2 \Big] - \frac{1}{2} \Big| \\
&= \Big| \Pr_{x \sim \mathcal{U}_n} \Big[\sum_{i \in S'} x_i \not\equiv f(x) \pmod 2 \Big] - \frac{1}{2} \Big| \\
&\stackrel{\text{def}}{=} p.
\end{aligned}$$

Let $f^- \colon \{-1,1\}^n \to \{-1,1\}$ be the corresponding version of f where we map 0 to 1, and 1 to -1, as usual. Observe that, under this correspondence,

$$\sum_{i \in S'} x_i \not\equiv f(x) \pmod 2 \quad \Longleftrightarrow \quad \chi_{S'}(x) \cdot f^-(x) = -1.$$

Therefore,

$$\begin{aligned}
p &= \Big| \Big(\frac{1}{2} - \frac{1}{2} \cdot \mathbb{E}_{x \sim \{-1,1\}^n} \big[\chi_{S'}(x) \cdot f^-(x) \big] \Big) - \frac{1}{2} \Big| \\
&= \Big| \Big(\frac{1}{2} - \frac{1}{2} \cdot \widehat{f^-}(S') \Big) - \frac{1}{2} \Big| \\
&= \frac{1}{2} \cdot |\widehat{f^-}(S')|.
\end{aligned}$$

In other words, in order to upper bound the distinguishing probability p, it is enough to upper bound $|\widehat{f^-}(S')|$, where $S' \subseteq [n]$. Using $f^-(x) = \prod_{i \in [k]} \mathsf{Tribes}^-_{n/k}(x^{(i)})$ and that $x^{(i)}$ and $x^{(j)}$ are disjoint for $i \neq j$, it follows that $\widehat{f^-}(S')$ is a product of Fourier coefficients of the corresponding Tribes functions. It is known that

$$\max_{T \subseteq [m]} \big| \widehat{\mathsf{Tribes}^-_m}(T) \big| = O\Big(\frac{\log m}{m} \Big)$$

as $m \to \infty$ (see e.g. O'Donnell [147]). Consequently, since we have $m = n/k$, we get that

$$p \;=\; \frac{1}{2} \cdot |\widehat{f^-}(S')| \;\leq\; \frac{1}{2} \cdot \max_{T \subseteq [n/k]} \big| \widehat{\mathsf{Tribes}^-_{n/k}}(T) \big|^k \;\leq\; \left(C \cdot (k/n) \cdot \log(n/k) \right)^k,$$

for an appropriate constant C. □

It is possible to use other monotone functions for the construction in Proposition 6.4.3, but our analysis provides better parameters with Tribes.

6.4.3 Pseudorandom functions

In this section we prove that a pseudorandon function is a highly non-monotone cryptographic primitive. For simplicity, we will not state the most general version of our result. We discuss some extensions after its proof.

Proposition 6.4.4. *If $F\colon \{0,1\}^m \times \{0,1\}^n \to \{0,1\}$ is a $(\mathsf{poly}(n), 1/3)$-secure pseudorandom function, then any Boolean circuit computing F contains at least $\log n - O(1)$ negation gates.*

Proof. Consider the following algorithm D^h that has membership access to an arbitrary function $h\colon \{0,1\}^n \to \{0,1\}$, and computes as follows. Let $\mathcal{X} \stackrel{\text{def}}{=} (e^0, e^1, \ldots, e^n)$ be the chain over $\{0,1\}^n$ with $e^i \stackrel{\text{def}}{=} 1^i 0^{n-i}$. After querying h on each input $e^0, \ldots, e^n$ and computing $a(h, \mathcal{X})$, D accepts h if and only if $a(h, \mathcal{X}) \geq n/4$. This completes the description of D. Clearly, this algorithm can be implemented in polynomial time.

Observe that if $f \sim \mathcal{F}_n$ is a random Boolean function over n variables, then $\mathbb{E}_f[a(f, \mathcal{X})] = n/2$. In addition, it follows from a standard application of Proposition 2.0.1 that $|a(f, \mathcal{X}) - n/2| \leq n/4$ with probability exponentially close to 1. Therefore, under our assumption that F is a $(\mathsf{poly}(n), 1/3)$-secure pseudorandom function,

$$\begin{aligned} \frac{1}{3} &\geq \left| \Pr_{w \sim \{0,1\}^n}[D^{F(w,\cdot)} = 1] - \Pr_{f \sim \mathcal{F}_n}[D^f = 1] \right| \\ &\geq \left| \Pr_{w \sim \{0,1\}^n}[D^{F(w,\cdot)} = 1] - (1 - o(1)) \right|, \end{aligned}$$

which implies in particular that $\Pr[D^{F(w,\cdot)} = 1] \geq 2/3 - o(1)$. Therefore, there must exist some seed w^* for which the resulting function $F_{w^*} \stackrel{\text{def}}{=} F(w^*, \cdot)$ over n-bit inputs satisfies $a(F_{w^*}, \mathcal{X}) \geq n/4$. It follows from Proposition 6.3.2 that if C is a circuit with t negations computing F_{w^*}, then

$$n/4 \;\leq\; a(F_{w^*}, \mathcal{X}) \;\leq\; a(F_{w^*}) \;\leq\; c \cdot 2^t,$$

where c is a fixed positive constant. Consequently, $t \geq \log n - O(1)$. Finally, it is clear that any circuit for F also requires $\log n - O(1)$ negations, which completes the proof. □

Note that we can replace $1/3$ with any constant in $[0, 1)$. The proof of Proposition 6.4.4 also implies that if F is a sufficiently secure pseudorandom function, then for most

choices of the seed $w \in \{0,1\}^m$, the resulting function $F(w,\cdot)$ over n input variables requires $\log n - O(1)$ negations. Further, observe that our distinguisher is quite simple, and makes $n+1$ *non-adaptive* queries.

The same proof does not work for *weak* pseudorandom functions. In this case, most random examples obtained from the oracle are concentrated around the middle layer of the hypercube, and one cannot construct a chain. We remark, however, that weak pseudorandom functions cannot be monotone, as there are weak learning algorithms for the class of monotone functions (cf. Blum, Burch, and Langford [31]). We discuss the problem of obtaining better lower bounds for WPRFs in Section 6.5. (The upper bound on the negation complexity of WPRFs follows via standard techniques, see Section 6.4.5 and Blais et al. [29].)

6.4.4 Error-correcting codes

In this section, we show that circuits with few negations cannot compute error-correcting codes with good parameters. The proof generalizes the argument given by Buresh-Oppenheim, Kabanets and Santhanam [44] in the case of *monotone* error-correcting codes.

Proposition 6.4.5. *Let $E\colon \{0,1\}^n \to \{0,1\}^m$ be an error-correcting code with relative distance $\gamma > 0$. If C is a circuit with t negations that computes E, then $t \geq \log n - \log(1/\gamma) - 1$.*

Proof. Assume that $E\colon \{0,1\}^n \to \{0,1\}^m$ is computed by a (multi-output) circuit C^0 with t negation gates, and let $x_1, \ldots, x_n$ be its input variables. For convenience, we write C^0_i to denote the Boolean function computed by the i-th output gate of C^0. We proceed as in the proof of Lemma 6.3.4. More precisely, we remove one negation gate during each step, but here we also inspect the behavior of the error-correcting code on a particular set of inputs of interest. Let $\mathcal{X} \stackrel{\text{def}}{=} (e^0, e^1, \ldots, e^n)$ be the chain over $\{0,1\}^n$ with $e^i \stackrel{\text{def}}{=} 1^i 0^{n-i}$.

It follows from an easy generalization of Lemma 6.3.5 that there exist Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$, $h\colon \{0,1\}^3 \to \{0,1\}$, and $g_{i,b}\colon \{0,1\}^n \to \{0,1\}$, where $i \in [m]$ and $b \in \{0,1\}$, for which the following holds.

- f is monotone;

- h is the addressing function $h(a, d_0, d_1) \stackrel{\text{def}}{=} d_a$;
- for every $x \in \{0,1\}^n$ and $i \in [m]$,

$$E(x)_i = h(f(x), g_{i,0}(x), g_{i,1}(x)).$$

- there exist (multi-output) circuits $C^{1,0}$ and $C^{1,1}$ over input variables $x_1, \ldots, x_n$ such that, for every $i \in [m]$ and $b \in \{0,1\}$,

$$C^{1,b}(x)_i = g_{i,b}(x).$$

- each circuit $C^{1,b}$ contains at most $t-1$ negations.

Since $e^0 \prec e^1 \prec \ldots \prec e^n$ and f is monotone, there exists $k \in [0,n]$ such that $f(e^\ell) = 0$ if and only if $\ell < k$. By the pigeonhole principle, f is constant on a (continuous) subchain $\mathcal{X}^1 \subseteq \mathcal{X}$ of size at least $(n+1)/2$, and there exists a constant $b \in \{0,1\}$ such that

$$E(e_i) = g_{1,b}(e^i) \ldots g_{m,b}(e^i),$$

whenever $e^i \in \mathcal{X}^1$. Consequently, there exists a (multi-output) circuit C^1 computed with at most $t-1$ negations that agrees with E on every $e^i \in \mathcal{X}^1$.

Observe that this argument can be applied once again with respect to $\mathcal{X}^1$ and C^1. Therefore, it is not hard to see that there must exist a chain $\mathcal{X}^t \subseteq \mathcal{X}$ of size $w \geq (n+1)/2^t$ and a *monotone* (multi-output) circuit C^t such that

$$C^t(e^i) = E(e^i),$$

for every $e^i \in \mathcal{X}^t$.

Assume that $\mathcal{X}^t = (e^j, e^{j+1}, \ldots, e^{j+w-1})$, and let $\mathcal{Y} \stackrel{\text{def}}{=} (y^j, \ldots, y^{j+w-1})$, where $y^i \stackrel{\text{def}}{=} E(e^i)$. Since C^t is *monotone* and $\mathcal{X}^t$ is a chain over $\{0,1\}^n$, we get that $\mathcal{Y}$ is a chain over $\{0,1\}^m$. By the pigeonhole principle, there exists an index $k \in [j+1, j+w-1]$ for which $y^{j-1} \preceq y^j$ and $|y^j|_1 - |y^{j-1}|_1 \leq (m+1)/w$. Now using that E computes an error-correcting code of relative distance at least γ, it follows that

$$\gamma \quad \leq \quad \Delta(y^j, y^{j-1}) \quad \leq \quad \frac{m+1}{w} \cdot \frac{1}{m} \quad \leq \quad \frac{2^t}{n+1} \cdot \frac{m+1}{m},$$

which completes the proof of our result. □

It is possible to show via a simple probabilistic construction that there is a sequence of error-correcting codes $E_n\colon \{0,1\}^n \to \{0,1\}^{O(n)}$ with relative distance, say, $\gamma = 0.01$ (see e.g. MacWilliams and Sloane [131]). Proposition 6.4.5 implies that computing such codes requires at least $\log n - O(1)$ negation gates, which is optimal up to the additive term via Markov's upper bound (Proposition 6.3.1).

6.4.5 Hard-core bits

We prove in this section that general hard-core predicates must be highly non-monotone. This result follows from a lower bound on the average-sensitivity of such functions due to Goldmann and Russell [78], together with structural results about monotone Boolean functions and Lemma 6.3.3. Roughly speaking, our result says that there are one-way functions that do not admit hardcore predicates computed with less than $(1/2) \cdot \log n$ negations (assuming that one-way functions exist).

Proposition 6.4.6. *Assume that there exists a family $f = \{f_n\}_{n\in\mathbb{N}}$ of $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions, where each $f_n\colon \{0,1\}^n \to \{0,1\}^n$. Then, for every $\varepsilon > 0$, there exists a family $g^\varepsilon = \{g_n\}_{n\in\mathbb{N}}$ of (length-preserving) $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions for which the following holds. If $h = \{h_n\}_{n\in\mathbb{N}}$ is a $(\mathsf{poly}(n), n^{-\omega(1)})$-secure hard-core bit for g^ε, then for every n sufficiently large, any Boolean circuit computing h_n contains at least $(1/2 - \varepsilon)\log n$ negations.*

Proof. It follows from the main result of Goldmann and Russell [78] that under the existence of one-way functions, there exists a one-way function family $g^\delta = \{g_n\}_{n\in\mathbb{N}}$ that only admits hard-core bit predicates with total influence $\Omega(n^{1-\delta})$. Our result follows easily once we observe that the influence of Boolean functions computed with t negations is $O(2^t \cdot \sqrt{n})$.[6]

First, if $f\colon \{0,1\}^n \to \{0,1\}$ is a monotone Boolean function, then $\mathrm{Inf}(f) = O(\sqrt{n})$ (see e.g. O'Donnell [147]). On the other hand, it follows from Lemma 6.3.3 that any Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ computed by a circuit with t negation gates can be written as $h(x) = P(m_1(x), \ldots, m_T(x))$, where $T = O(2^t)$, each function m_i is monotone, and P is

[6] This result is from Blais et al. [29], and we include its short argument here for completeness.

either the parity function or its negation. Therefore, using the definition of influence,

$$\mathrm{Inf}(h) \;=\; \mathrm{Inf}(P(m_1,\ldots,m_T)) \;\le\; \sum_{i\in[T]} \mathrm{Inf}(m_i) \;\le\; T\cdot O(\sqrt{n}) \;=\; O(2^t\cdot\sqrt{n}),$$

which completes the proof. □

This result is almost optimal, as any function $f\colon \{0,1\}^n \to \{0,1\}$ can be $(1/n^{\omega(1)})$-approximated by a Boolean function computed with $(1/2+o(1))\log n$ negations (check Blais et al. [29] for more details). More precisely, if h is a hard-core bit for f, its approximator $\widetilde{h}$ is also hard-core for f, as the inputs $f(x)$ given to the distinguisher are produced with $x \sim \mathcal{U}_n$.

6.4.6 Randomness extractors

In this section, we show in Proposition 6.4.8 that strong $(n^{0.5-\varepsilon}, 1/2)$-extractors can only be computed by circuits with $\Omega(\log n)$ negation gates, for any constant $0 < \varepsilon \le 1/2$. We proceed as follows. First, we argue that such extractors must have high noise sensitivity. The proof of this result employs a technique from Bogdanov and Guo [36]. We then upper bound the noise sensitivity of circuits with few negations. Together, these claims provide a trade-off between the parameters of the extractor, and the minimum number of negations in any circuit computing the extractor.

For convenience, we view the extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ as a family of functions

$$\mathcal{H}_{\mathsf{Ext}} \stackrel{\text{def}}{=} \{h_w\colon \{0,1\}^n \to \{0,1\}^m \mid h_w = \mathsf{Ext}(\cdot, w), \text{ where } w \in \{0,1\}^s\},$$

i.e., the family of functions obtained from the extractor by fixing its seed. Similarly, every such family can be viewed as a strong extractor in the natural way.

Lemma 6.4.7. *Let $0 \le p \le 1/2$, $0 \le \gamma \le 1/4$, and $\mathcal{H} \subseteq \{h \mid h\colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of functions. Assume that $\mathsf{NS}_p(h_i) \le \gamma$ for every function $h_i\colon \{0,1\}^n \to \{0,1\}$ that computes the i-th output bit of some function in $\mathcal{H}$, where $i \in [m]$. Then there exists a distribution D over $\{0,1\}^n$ with min-entropy $H_\infty(D) = n\cdot\log(\frac{1}{1-p})$ such that the statistical distance between $(\mathcal{H}, \mathcal{H}(D))$ and $(\mathcal{H}, \mathcal{U}_m)$ is at least $(1-2\sqrt{\gamma}-2^{-0.1m})(1-2\sqrt{\gamma})$.*

Proof. For a fixed $y \in \{0,1\}^n$, let D_y denote a random variable distributed according to $y \oplus X$, where X is the p-biased binomial distribution over $\{0,1\}^n$. Since $p \leq 1/2$, observe that the min-entropy of D_y is precisely

$$\begin{aligned} H_\infty(D_y) &= -\log \max_{z \in \{0,1\}^n} \Pr[y \oplus X = z] = -\log \Pr[y \oplus X = y] \\ &= -\log (1-p)^n = n \cdot \log\left(\frac{1}{1-p}\right). \end{aligned}$$

We will need the following result.

Claim. For any fixed $h \in \mathcal{H}$,

$$\mathrm{E}_{y \sim \{0,1\}^n}[\delta(h(D_y), \mathcal{U}_m)] \geq (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}). \tag{6.2}$$

We use this claim to complete the demonstration of Lemma 6.4.7, then return to its proof. Observe that, for any fixed $y \in \{0,1\}^n$,

$$\delta((\mathcal{H}, \mathcal{H}(D_y)), (\mathcal{H}, \mathcal{U}_m)) = \mathrm{E}_{h \sim \mathcal{H}}[\delta(h(D_y), \mathcal{U}_m)]. \tag{6.3}$$

It follows from Equation 6.3 that

$$\begin{aligned} \mathrm{E}_{y \sim \{0,1\}^n}[\delta((\mathcal{H}, \mathcal{H}(D_y)), (\mathcal{H}, \mathcal{U}_m))] &= \mathrm{E}_{y \sim \{0,1\}^n}[\mathrm{E}_{h \sim \mathcal{H}}[\delta(h(D_y), \mathcal{U}_m)]] \\ &= \mathrm{E}_h[\mathrm{E}_y[\delta(h(D_y), \mathcal{U}_m)]] \\ \text{(Using Equation 6.2)} \quad &\geq \mathrm{E}_h[(1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma})] \\ &= (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}). \end{aligned}$$

In particular, there exists $y \in \{0,1\}^n$ such that

$$\delta((\mathcal{H}, \mathcal{H}(D_y)), (\mathcal{H}, \mathcal{U}_m)) \geq (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}),$$

which completes the proof of Lemma 6.4.7.

We proceed now to the proof of our initial claim. Fix a function $h \in \mathcal{H}$. By the definition of noise sensitivity and our assumption on $\mathcal{H}$, for every function $h_i \colon \{0,1\}^n \to \{0,1\}$ obtained from a function $h \in \mathcal{H}$ as the projection of the i-th output bit, we have

$$\Pr_y[h_i(D_y) \neq h_i(y)] = \Pr_y[h_i(y \oplus X) \neq h_i(y)] \leq \gamma.$$

Using the linearity of expectation, we obtain

$$\mathrm{E}_y[\Delta(h(D_y), h(y))] \leq \gamma.$$

By Markov's inequality,

$$\Pr_y[\Delta(h(D_y), h(y)) \leq 1/4] \geq 1 - 4\gamma.$$

Using an averaging argument, with probability at least $1 - \sqrt{4\gamma}$ over the choice of y, we have that

$$\Pr[\Delta(h(D_y), h(y)) \leq 1/4] \geq 1 - \sqrt{4\gamma}. \tag{6.4}$$

For any fixed y, consider the following statistical test,

$$T_y \stackrel{\text{def}}{=} \{z \in \{0,1\}^m \mid \Delta(z, h(y)) \leq 1/4\}.$$

The probability that $\mathcal{U}_m \in T_y$ can be upper bounded via a standard inequality by

$$\Pr_{z \sim \mathcal{U}_m}[z \in T_y] \leq \frac{2^{m \cdot H_2(1/4)}}{2^m} \leq 2^{-0.1m}, \tag{6.5}$$

where $H_2 \colon [0,1] \to [0,1]$ is the binary entropy function, and we use the fact that $H_2(1/4) \leq 0.9$. Combining Equations 6.4 and 6.5, we get

$$\Pr_y\left[(\Pr_X[h(D_y) \in T_y] - \Pr_{z \sim \mathcal{U}_m}[z \in T_y]) \geq 1 - \sqrt{4\gamma} - 2^{-0.1m}\right] \geq 1 - \sqrt{4\gamma},$$

which implies that

$$\Pr_y[\delta(h(D_y), \mathcal{U}_m) \geq 1 - 2\sqrt{\gamma} - 2^{-0.1m}] \geq 1 - 2\sqrt{\gamma}.$$

Finally, since $\delta(\cdot)$ is non-negative and $\gamma \leq 1/4$, it follows that

$$\mathrm{E}_y[\delta(h(D_y), \mathcal{U}_m)] \geq (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}),$$

which completes the proof of the claim. □

We are now ready to prove a lower bound on the negation complexity of strong extractors.

Proposition 6.4.8. *Let $0 < \alpha < 1/2$ be a constant, and $m(n) \geq 100$. Further, suppose that $\mathcal{H} \subseteq \{h \mid h\colon \{0,1\}^n \to \{0,1\}^m\}$ is a family of functions such that each output bit $h_i\colon \{0,1\}^n \to \{0,1\}$ of a function $h \in \mathcal{H}$ is computed by a circuit with t negations. Then, if $\mathcal{H}$ is an $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor,*

$$t \;\geq\; \alpha \log n - O(1).$$

Proof. It is known that for any monotone function $g\colon \{0,1\}^n \to \{0,1\}$ and $p(n) \in (0,1/2)$, $\mathsf{NS}_p(g) = O(\sqrt{n} \cdot p)$ (see e.g. O'Donnell [147]). Using an argument similar to the one employed in the proof of Proposition 6.4.6, it follows from Lemma 6.3.3 that if $f\colon \{0,1\}^n \to \{0,1\}$ is a Boolean function computed by a circuit with t negations, then

$$\mathsf{NS}_p(f) \;\leq\; C_1 \cdot 2^t \sqrt{n} \cdot p \;\stackrel{\text{def}}{=}\; \gamma,$$

where $C_1 > 0$ is a fixed constant. In other words, this upper bound on the noise sensitivity and our assumption on $\mathcal{H}$ allow us to apply Lemma 6.4.7 with an appropriate choice of parameters, which we describe next.

We choose a $0 \leq p \leq \frac{1}{2}$ such that $n \cdot \log \frac{1}{(1-p)} = n^{\frac{1}{2}-\alpha}$. Observe that we can take $p \leq C_2 n^{-\frac{1}{2}-\alpha}$, for an appropriate constant $C_2 > 0$. Let C_3 be a sufficiently large constant such that $C_1 C_2 2^{-C_3} < 1/64$, and suppose that $t < \alpha \log n - C_3$. For this setting of parameters, we obtain

$$\gamma \;=\; C_1 \cdot 2^t \cdot \sqrt{n} \cdot p \;<\; \frac{1}{64}.$$

By Lemma 6.4.7, there exists a distribution D of min-entropy $H_\infty(D) = n \log \frac{1}{1-p} = n^{\frac{1}{2}-\alpha}$ for which

$$\begin{aligned} \delta((\mathcal{H}, \mathcal{H}(D)), (\mathcal{H}, \mathcal{U}_m)) \;&\geq\; (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}) \\ &>\; (\frac{3}{4} - 2^{-0.1m}) \cdot \frac{3}{4} \;\geq\; \frac{1}{2}, \end{aligned}$$

which contradicts our assumption that $\mathcal{H}$ is an $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor. Therefore,

$$t \;\geq\; \alpha \log n - C_3 \;=\; \alpha \log n - O(1),$$

as desired. □

Observe that Proposition 6.4.8 provides an almost tight lower bound on the number of negations for extractors with rather weak parameters: in order to extract from reasonable sources only 100 bits that are not ridiculously far from uniform, the corresponding circuits need $\Omega(\log n)$ negations.

6.4.7 Negations at the bottom layer and circuit lower bounds

In this section we investigate the power of a restricted number of negations at the bottom layer. As discussed in Section 6.4.7, our proof relies on an idea from Koroth and Sarma [124]. Our main contribution is the following general proposition.

Proposition 6.4.9. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a monotone Boolean function, and C be a circuit computing f with negation gates at the bottom layer only. Then,*

$$\mathsf{depth}(C) + \mathsf{negations}(C) \geq \mathsf{depth}^{+}(f).$$

Proof. Let $d \stackrel{\text{def}}{=} \mathsf{depth}(C)$, and $t \stackrel{\text{def}}{=} \mathsf{negations}(C)$. The idea is to use C, a non-monotone circuit for f, to solve the corresponding monotone Karchmer-Wigderson game of f with communication at most $d + t$. It follows from Proposition 2.0.2 that $\mathsf{depth}^{+}(f) \leq d + t$, which completes the proof. More details follow.

Recall that in the monotone Karchmer-Wigderson game for f, Alice is given a string $x \in f^{-1}(1)$, Bob is given $y \in f^{-1}(0)$, and their goal is to agree on a coordinate i such that $x_i = 1$ and $y_i = 0$. Let $T \subset [n]$ be the set of variables that occur negated in C, where $|T| = t$. Given a string $x \in \{0,1\}^n$, we write x_T to denote the substring of x obtained by concatenating the bits indexed by T. During the first round of the protocol, Alice sends x_T to Bob. If among these coordinates there is an index $i \in T$ for which $x_i = 1$ and $y_i = 0$, the protocol terminates with a correct solution. Otherwise, Bob defines a new input $y' \in \{0,1\}^n$ for him as follows: $y'_j \stackrel{\text{def}}{=} x_j$ if $j \in T$, otherwise $y'_j \stackrel{\text{def}}{=} y_j$. For convenience, Alice sets $x' \stackrel{\text{def}}{=} x$.

It is not hard to see that if there was no good index $i \in T$, then $f(x') = 1$ and $f(y') = 0$. Clearly, $1 = f(x) = f(x')$, since $x = x'$. On the other hand, if there is no good index i, y' is obtained from y simply by flipping some bits of y from 1 to 0. In other words, $y' \preceq y$, and the monotonicity of f implies that $f(y') \leq f(y) = 0$.

Crucially, the players now have inputs $x', y' \in \{0,1\}^n$ that agree on every bit indexed by T. Therefore, without any communication, they are able to simplify the original circuit C in order to obtain a *monotone* circuit $\tilde{C}$ with input variables indexed by $[n]\backslash T$. Let $\tilde{x} \stackrel{\text{def}}{=} x'_{[n]\backslash T}$ and $\tilde{y} \stackrel{\text{def}}{=} y'_{[n]\backslash T}$ be the corresponding projections of x' and y'. Clearly, $\tilde{C}(\tilde{x}) = C(x') = f(x') = 1$, and $\tilde{C}(\tilde{y}) = C(y') = f(y') = 0$. Furthermore, $\tilde{C}$ computes some monotone function $\tilde{f}\colon \{0,1\}^{[n]\backslash T} \to \{0,1\}$.

Alice and Bob simulate together the standard Karchmer-Wigderson protocol Π granted by Proposition 2.0.2, and obtain an index $j \in [n]\backslash T$ for which $\tilde{x}_j = 1$ and $\tilde{y}_j = 0$. Observe that this stage can be executed with communication cost $\mathsf{depth}(\tilde{C}) \leq \mathsf{depth}(C) = d$. However, since x agrees with $\tilde{x}$ on every bit indexed by $[n]\backslash T$, and similarly for y and $\tilde{y}$, it follows that $x_j = 1$ and $y_j = 0$. Put another way, Alice and Bob have solved the monotone Karchmer-Wigderson game for f with communication at most $t + d$, which completes the proof of our result. □

An interesting aspect of this proof is that it relies on both directions of the Karchmer-Wigderson connection. Proposition 6.4.9 and previous work on monotone depth lower bounds provide a trade-off between circuit depth and negation complexity for DeMorgan circuits solving the clique problem.

Proposition 6.4.10 (Raz and Wigderson [153])**.** *Let k-*Clique*$\colon \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be the Boolean function that is 1 if and only if the input graph G contains a clique of size k. If C is a monotone circuit that computes k-*Clique *for $k = n/2$, then $\mathsf{depth}(C) \geq \gamma \cdot n$, where $\gamma > 0$ is a fixed constant.*

Corollary 6.4.11. *There exists a fixed constant $\gamma > 0$ for which the following holds. If $\delta + \varepsilon \leq \gamma$, then any DeMorgan circuit of depth δn solving the $(n/2)$-*Clique *problem on n-vertex graphs contains at least εn negation gates.*

This result indicates that negation gates at the bottom layer are much easier to handle from the point of view of complexity theory than negations located at arbitrary positions of the circuit (see also Proposition 6.6.2 in Section 6.6).

6.5 Open problems and further research directions

While our results provide some strong bounds, they also leave open surprisingly basic questions.

For example, it seems reasonable, in light of our results, to think that most cryptographic primitives require $\Omega(\log n)$ negations. Nevertheless, for a basic primitive like a pseudorandom generator (that cannot be monotone), we leave open the following question: Is there a pseudorandom generator computed with a single negation gate? We stress that our question refers to a single circuit with multiple output bits computing the PRG. If one can use different circuits for distinct output bits, then the work of Applebaum, Ishai, and Kushilevitz [18] provides strong evidence that there are PRGs computed with a constant number of negations.

Having negation gates at the bottom level may be easier to study, and with some work we can show (in results omitted from this thesis) that no function with large enough stretch computed with a single negation at the bottom layer can be a small-bias generator (and thus not a pseudorandom generator either).

Another important open problem relates to the negation complexity of WPRF (weak pseudorandom functions, cf. Akavia et al. [9]), or, viewed from the learning perspective, weak-learning functions computed with a single negation. While for strong PRFs, even non-adaptive ones, we have obtained an $\Omega(\log n)$ lower bound, as far as we know, there may exist WPRFs computed by circuits with a single negation gate. Again, when restricting ourselves to negations at the bottom, we can prove some partial results (it is not hard to prove that a function computed by a circuit with a constant number of negations at the bottom layer cannot be a WPRF).

Finally, we have not imposed additional restrictions on the structure of Boolean functions computing cryptographic primitives. For instance, due to efficiency concerns, it is desirable that such circuits have depth as small as possible, without compromising the security of the underlying primitive. It is known that Markov's upper bound of $O(\log n)$ negations fails under restrictions of this form (cf. Santha and Wilson [165]; see also Hofmeister [97]). In particular, this situation sheds some light into why practical implementations have far more negations (or XORs) when compared to the theoretical lower bounds described in our

work. Here we have not investigated this phenomenon, and it would be interesting to see if more specific results can be obtained in the cryptographic context.

6.6 Auxiliary results

In this section we discuss how to move negations in a Boolean circuit in order to explore different aspects of these gates.

6.6.1 Moving negations to the top of the circuit

Recall the following structural result about negation gates, mentioned in Section 6.3.

Lemma (Blais et al. [29])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C with at most t negations. Then f can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is monotone, $T = O(2^t)$, and h is either the parity function, or its negation.*

By relaxing the assumption on h, we can prove the following effective version of Lemma 6.3.3.

Lemma. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C of size s containing t negation gates, where $t \geq 0$. Then f can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$, where each function g_i is computed by a monotone circuit of size at most s, $T = 2^{t+1} - 1$, and $h\colon \{0,1\}^T \to \{0,1\}$ is computed by a circuit of size at most $5T$.*

Proof. The proof is by induction on t. The base case $t = 0$ is trivial. Now let $t \geq 1$, and assume the statement holds for any function computed by circuits with at most $t' < t$ negations. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a circuit C of size s that contains t negations. Let $x_1, \ldots x_n, f_1, \ldots, f_s$ be the functions computed at each internal node of C, and $\lceil f_1 \rceil, \ldots, \lceil f_s \rceil$ be the corresponding gates, i.e., each $\lceil f_i \rceil \in \{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$. Furthermore, assume that this sequence is a topological sort of the nodes of the circuit, in the sense that the inputs of each gate $\lceil f_i \rceil$ are $f_{i_1}(x)$ and $f_{i_2}(x)$, with $i_1, i_2 < i$. Let $i^* \in [s]$ be the index of the first NOT gate in C according to this sequence.

Consider a new circuit C' over $n+1$ variables $x_1, \ldots, x_n, y$, where C' computes exactly as C, except that the output value of f_{i^*} is replaced by the new input y. By construction, C' is a circuit of size at most s containing $t' \stackrel{\text{def}}{=} t-1$ negations, and it computes some Boolean function $f' \colon \{0,1\}^{n+1} \to \{0,1\}$. Applying the induction hypothesis, we get that

$$f'(\vec{x}, y) = h'(g'_1(\vec{x}, y), \ldots, g'_{T'}(\vec{x}, y)), \tag{6.6}$$

where each g'_i is computed by a monotone circuit of size at most s, $T' \leq 2^{t'+1} - 1$, and $h' \colon \{0,1\}^{T'} \to \{0,1\}$ admits a circuit of size $5T'$. In addition, notice that

$$f(\vec{x}) = \begin{cases} f'(\vec{x}, 1) & \text{if} \quad f_{i^*}(\vec{x}) = 1, \\ f'(\vec{x}, 0) & \text{otherwise.} \end{cases} \tag{6.7}$$

Let f_i be the input wire of $\lceil f_{i^*} \rceil$. Since $\lceil f_{i^*} \rceil = \mathsf{NOT}$, we obtain using Equation 6.7 that

$$f(\vec{x}) = \tilde{h}(f_i(\vec{x}), f'(\vec{x}, 0), f'(\vec{x}, 1)), \tag{6.8}$$

where $\tilde{h}(z, y_1, y_0) \stackrel{\text{def}}{=} y_z$ is a function over three input bits that is computed by a circuit of size at most 5. Furthermore, combining Equations 6.6 and 6.8, it follows that

$$\begin{aligned} f(\vec{x}) &= \tilde{h}(f_i(\vec{x}), h'(g'_1(\vec{x}, 0), \ldots, g'_{T'}(\vec{x}, 0)), h'(g'_1(\vec{x}, 1), \ldots, g'_{T'}(\vec{x}, 1))) \\ &= h(f_i(\vec{x}), g'_{1,0}(\vec{x}), \ldots, g'_{T',0}(\vec{x}), g'_{1,1}(\vec{x}), \ldots, g'_{T',1}(\vec{x})), \end{aligned}$$

where $g'_{j,b}(\vec{x}) \stackrel{\text{def}}{=} g'_j(\vec{x}, b)$, for $j \in [T']$ and $b \in \{0,1\}$, and $h \colon \{0,1\}^{2T'+1} \to \{0,1\}$ is the function obtained by setting

$$h(v_0, v_1, \ldots, v_{T'}, v_{T'+1}, \ldots, v_{2T'}) \stackrel{\text{def}}{=} \tilde{h}(v_0, h'(v_1, \ldots, v_{T'}), h(v_{T'+1}, \ldots, v_{2T'})).$$

Using our assumption on i^*, it follows that f_i is computed by a monotone circuit of size s. It is also clear that each $g'_{j,b}$ admits a monotone circuit of size s. Further, observe that

$$2T' + 1 \leq 2(2^{t'+1} - 1) + 1 = 2(2^t - 1) + 1 = 2^{t+1} - 1 \stackrel{\text{def}}{=} T.$$

Finally, using the induction hypothesis and the upper bound on the circuit size of $\tilde{h}$, we get that h is computed by a circuit of size at most

$$5 + 5T' + 5T' = 5(2T' + 1) = 5T,$$

which completes the proof of Lemma 6.3.4. □

It is possible to show that the upper bound on T in the statement of Lemma 6.7 is essentially optimal. This follows from the connection between the number of negation gates in a Boolean circuit for a function f and the alternation complexity of f, as discovered by Markov [132] (see e.g. Blais et al. [29] for further details).

6.6.2 Moving negations to the bottom of the circuit

We recall the following basic fact about negations.

Fact 6.6.1. *Let C be a Boolean circuit of size s containing a negation gate at depth $d \geq 1$. Then C can be transformed into an equivalent circuit C' of size s without this negation gate that contains at most 2^{d-1} additional negations at the bottom layer.*

Proof. The result is immediate from the application of DeMorgan rules for Boolean connectives. □

We observe below that this result is optimal. Put another way, a negation gate at an arbitrary location can be more powerful than a linear number of negations at the bottom layer.

Proposition 6.6.2. *There exists an explicit Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ that admits a linear size circuit C containing a single negation gate, but for which any equivalent circuit C' with negation gates at the bottom layer only requires n negations.*

Proof. Let $f(x) = 1$ if and only if $x = 0^n$. Clearly, f can be computed by a circuit with a single negation, since this function is the negation of a monotone function. The lower bound follows using an argument from [124]. Assume that $f(x) = D(x, (x \oplus \beta))$, where D is a monotone circuit. We need to prove that $\beta_i = 1$ for every $i \in [n]$. Consider inputs $z \stackrel{\text{def}}{=} 0^n$ and $e_i \stackrel{\text{def}}{=} 0^{i-1}10^{n-i}$. By definition, $f(z) = 1$ and $f(e_i) = 0$, thus $D(0^n, 0^n \oplus \beta) = D(0^n, \beta) = 1$ and $D(e_i, e_i \oplus \beta) = D(e_i, \beta^{\oplus i}) = 0$. If $\beta_i = 0$, then $(0^n, \beta) \prec (e_i, \beta^{\oplus i})$, and since D is monotone, we get $D(0^n, \beta) \leq D(e_i, \beta^{\oplus i})$. However, this is in contradiction with the value of f on z and e_i, which implies that $\beta_i = 1$. □

Part III

Connections between Algorithms and Circuit Lower Bounds

Chapter 7

Constructing hard functions from learning algorithms

7.1 Background, results, and organization

Understanding the computational complexity of learning circuit classes continues to be an important area of study in theoretical computer science. For example, recent work of Gentry and Halevi [77] makes use of results on the complexity of learning depth-3 arithmetic circuits [121] to construct improved homomorphic encryption schemes. More generally, the relationship between the complexity of learning circuits and cryptography has been extensively studied over the last twenty years (e.g., [193] and [116]).

Less is known regarding the relationship between learning circuit classes and proving circuit lower bounds. Historically, circuit lower bounds for a class $\mathcal{C}$ typically precede the design of a learning algorithm for $\mathcal{C}$. Some intuition for this fact is that circuit lower bounds usually reveal important structural properties of these circuit classes, allowing them to be learned in some non-trivial way.

Fortnow and Klivans [70] were the first to codify this intuition and prove formally that efficient learning algorithms (in a variety of models) for a circuit class $\mathcal{C}$ yield circuit lower bounds against $\mathcal{C}$. Their result reduces the task of proving circuit lower bounds to the task of designing efficient learning algorithms. They showed, for example, that a polynomial-time PAC learning algorithm for $\mathcal{C}$ separates BPEXP from $\mathcal{C}$. Additionally

they proved that a subexponential time *exact* learning algorithm separates $\mathsf{EXP}^{\mathsf{NP}}$ from $\mathcal{C}$ (this was subsequently improved to EXP by Hitchcock and Harkins [92] using techniques from resource bounded measure). Their proof uses a variety of classic complexity-theoretic results such as Toda's theorem, the complexity of the Permanent, collapse theorems ($\mathsf{EXP} \subseteq \mathsf{P/poly} \Rightarrow \mathsf{EXP} = \mathsf{MA}$ [23]), and hierarchy theorems.

In this chapter we prove that it is possible to derive stronger circuit lower bounds from learning algorithms. Our results significantly improve and expand on the above initial work by Fortnow and Klivans. In many cases our proofs are simple, self-contained, and do not use machinery from computational complexity. We obtain these consequences in a variety of well-known learning models: PAC, online (mistake-bounded), exact, and statistical query learning. We begin by outlining our main results and contrasting them with previous work.

7.1.1 Improved separations for Online and Exact Learning

Our first set of results deals with learning algorithms in the online (mistake-bounded) model of learning and Angluin's model of exact learning with membership and equivalence queries. Recall that in the mistake-bounded model of learning, a function c from some class $\mathcal{C}$ is fixed, and a learner is sequentially presented with an arbitrary sequence of examples. After receiving example x_i, the learner must output its prediction for $c(x_i)$. We say that a learner succeeds with mistake bound m if for any (possibly infinite) sequence of examples, the learner makes at most m mistakes. The time-complexity $T(n, s)$ of the learner is the amount of time taken when presented with an example of length n and when c has size at most s. We prove the following theorem relating mistake-bounded learning to circuit lower bounds:

Theorem 7.1.1. *Let $\mathcal{C}^s$ be a non-uniform class of circuits where each $c \in \mathcal{C}^s$ has size at most s (according to some fixed representation). If $\mathcal{C}^s$ is learnable in the mistake-bounded model in time $T = T(n, s)$ and mistake bound $M = M(n, s) < 2^n$, then there exists an explicit function f computable in $\mathsf{DTIME}(M \cdot T)$ such that $f \notin \mathcal{C}^s$.*

Our proof actually shows that f is $\Omega(1/M)$-far from every $c \in \mathcal{C}$. For the class of polynomial-size circuits, the above theorem yields new circuit lower bounds as long as the

learning algorithm has non-trivial run-time and mistake bound. If the learning algorithm is efficient (polynomial run-time and mistake bound) we obtain the following corollary:

Corollary 7.1.2. *Let $\mathcal{C}$ be any class of polynomial-size circuits (e.g., $\mathsf{AC}^0, \mathsf{TC}^0, \mathsf{P/poly}$). If $\mathcal{C}$ is efficiently learnable in the mistake-bounded model then $\mathsf{DTIME}(n^{\omega(1)}) \not\subseteq \mathcal{C}$.*

With more work, we prove analogous results for Angluin's model of exact learning with membership and equivalence queries. In this model the learner is required to exactly learn the target function c, and is allowed to query the value $c(x)$ on any input x (membership query). The learning algorithm can also check if a proposed hypothesis h is equivalent to c (equivalence query). If this is not the case, it is presented with a counterexample w for which $h(w) \neq c(w)$. It is not hard to see that learnability in the mistake-bounded model implies learnability in Angluin's model.

Previous work due to Fortnow and Klivans [70] and also Hitchcock and Harkins [92] proved, under a learning assumption, the existence of a hard function for polynomial-size circuits in $\mathsf{EXP}^{\mathsf{NP}}$ and EXP, respectively. In contrast, our proof yields an explicit function that is computable in any superpolynomial time class. Since we are able to explicitly construct hard functions in lower (uniform) deterministic time complexity classes (recall that our learning algorithms are assumed to be deterministic), we can prove that efficient learning algorithms imply a full derandomization of BPP:

Corollary 7.1.3. *Let $\mathcal{C}$ be the class of linear-size circuits. If $\mathcal{C}$ is efficiently exactly learnable (or learnable in the mistake-bounded model) then $\mathsf{P} = \mathsf{BPP}$.*

Our results for mistake-bounded and exact learning use the learning algorithms themselves in non-standard ways to construct hard functions. For example, in a typical learning scenario, the learning algorithm receives examples all of which are labelled according to some fixed function $c \in \mathcal{C}$. In our setting, we will run our mistake-bounded or exact learning algorithms in stages, using *different functions* to provide labels for the examples in each stage. More specifically, we will continually label new examples according to the *negation* of the learning algorithm's current hypothesis. At first glance, it would seem that no guarantees can be made about a learning algorithm that is not given examples labelled according to a fixed function. Still, we are able to use the learning algorithm to "fool" all

potential functions that it might have to learn. At a high level, we consider this a sort of diagonalization over all elements of $\mathcal{C}$. We give more details on this procedure in Section 7.3.

In contrast, the work of Fortnow and Klivans is considerably more complicated, requiring non-trivial collapse arguments, hierarchy theorems, Toda's theorem, and various well-known properties of the Permanent function in order to obtain their conclusion. Hitchcock and Harkins used betting games and ideas from resource bounded measure to obtain their improvement. As can be seen in Section 7.3, our proof is simple, self-contained, and yields a much finer separation. We note that the same proof was discovered independently by Impagliazzo and Kabanets [112].

7.1.2 Hard functions in PSPACE

The previous set of results showed that *deterministic* learning algorithms in the exact or mistake-bounded model imply hard functions computable in subexponential-time uniform complexity classes. We also investigate the possibility of constructing hard functions in PSPACE, given the existence of non-trivial *randomized* learning algorithms. We prove that unless PSPACE lies in randomized sub-exponential time, non-trivial learning algorithms in the PAC model imply the existence of hard functions in PSPACE. Actually, this is true even if the PAC learning algorithm is allowed membership queries and only works under the uniform distribution:

Theorem 7.1.4. *Let $\mathcal{C}$ be any circuit class and suppose that there exists a randomized algorithm that* PAC *learns $\mathcal{C}$ under the uniform distribution using membership queries in time $O(T(n, \mathsf{size}(c)))$, where $c \in \mathcal{C}$ is the unknown concept. Then, for any function $s : \mathbb{N} \to \mathbb{N}$, at least one of the following conditions hold:*

(i) There are languages in PSPACE not computed by circuits from $\mathcal{C}$ of size s; or

(ii) $\mathsf{PSPACE} \subseteq \mathsf{BPTIME}(T(n, s))$.

In contrast, Fortnow and Klivans proved that for any circuit class $\mathcal{C} \subseteq \mathsf{P/poly}$, if $\mathcal{C}$ is PAC learnable under the uniform distribution by a polynomial-time algorithm with membership queries then $\mathsf{BPEXP} \not\subseteq \mathcal{C}$. Theorem 7.1.4 extends their work in the following

directions: (*i*) we obtain interesting consequences for PSPACE instead of BPEXP; (*ii*) it is possible to derive new results for PSPACE even in the case that the learning algorithm does not run in polynomial time; (*iii*) $\mathcal{C}$ does not need to be contained in P/poly, which means that this result can (under the learning assumptions) be used to obtain super-polynomial lower bounds. In Section 7.4, we explain how Fortnow and Klivans's original result can be derived from Theorem 7.1.4.

Note that the second condition in the conclusion of this theorem does not depend on the original circuit class that appears in the hypothesis. While this seems odd at first, we give a simple proof that removing this "or" condition from the conclusion of the theorem would give us an unconditional proof that PSPACE $\not\subseteq$ BPP. In other words, proving strong theorems of the form "learning implies circuit lower bounds" yields important uniform separations.

Theorem 7.1.4 also explains the difficulty of designing non-trivial PAC learning algorithms for the class of polynomial-size depth-two threshold functions, also known as TC^0_2. This is one of the smallest circuit classes for which there are no known non-trivial circuit lower bounds. In particular, it could be the case that any language in BPEXP is in TC^0_2. Our result shows that the existence of a non-trivial PAC learning algorithm for this class provides strong evidence that PSPACE does not admit such circuits. Previous results required stronger assumptions. For instance, the original theorem proven by Fortnow and Klivans [70] assumes efficient PAC learnability, and the cryptographic hardness results of Klivans and Sherstov [121] do not hold with respect to the uniform distribution or when learner is allowed membership queries.

The main idea of the proof is to rework the Fortnow and Klivans approach but use a PSPACE-complete problem described by Trevisan and Vadhan [187] that is downward self-reducible and self-correctible. In contrast, Fortnow and Klivans used the Permanent function (and its well-known self-reducibility properties) but had to first go through a "collapse" argument to arrive in a scenario where the Permanent is complete for PSPACE. The proof of Theorem 7.1.4 is presented in Section 7.4.

We also observe that Karp-Lipton style collapse results follow easily from a relativized version of Theorem 7.1.4 and Occam's Razor (Blumer et al. [32]), for any complexity

class with complete problems that are both downward self-reducible and self-correctible. While learning theory techniques have been used to prove collapse theorems in the past (cf. Bshouty et al. [41]), our argument based on Occam's Razor seems to be new.

7.1.3 Average-case hard functions from Statistical Query Learning

Our results above show that nontrivial learning algorithms in the exact, mistake-bounded, or PAC model yield functions that are hard to compute in the worst-case. We show that even weak learning algorithms that use only *Correlational Statistical Queries* (CSQs) yield not just circuit lower bounds but explicit functions that are hard to compute *on average*. Informally, a CSQ learner is allowed to make queries of the form $\mathrm{E}[q \cdot c]$ where c is the target function and q is some fixed (polynomial-time computable) predicate. The learner then receives an estimate of the true value of this query to within τ, a "tolerance" parameter. CSQ learning has been an active area of study recently in computational learning theory. It is known [64] that the class of functions that are Evolvable (in the sense of Valiant [194]) are exactly equal to the functions that are learnable by CSQs (we define Correlational Statistical Queries formally in Section 7.2.4). We give the following consequence for CSQ learning:

Theorem 7.1.5. *Let $\mathcal{C}$ be a representation class of Boolean functions on $\{-1,1\}^n$. Let ϵ, τ be any parameters satisfying $\epsilon < \frac{1}{2}$ and $2^{-o(n)} \leq \tau \leq \min\{\epsilon, 1-2\epsilon\}$. Suppose there exists an algorithm $\mathcal{A}$ that runs in time $T = T(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ that learns $\mathcal{C}^s$ on the uniform distribution in the CSQ model to accuracy $1-\epsilon$ by at most $Q = Q(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s) \leq 2^n$ queries, each of tolerance τ. Then, there exists a Boolean function (family) $f \in \mathsf{DTIME}(T + \mathrm{poly}(Q, \frac{1}{\tau}))$ such that for every $c \in \mathcal{C}^s$, $\Pr_{x \sim \mathcal{U}}[f(x) \neq c(x)] \geq \frac{\tau}{4}$.*

We note that a weak average-case hardness for an explicit function (parity) can be obtained by a simple argument based on SQ-dimension [30]. For example, it follows from the definition of SQ-dimension that if $\mathcal{C}$ has polynomial SQ-dimension any $c \in \mathcal{C}$ differs from parity on a non-zero but negligible fraction of inputs (this is discussed in more detail in Section 7.5.2). Since τ is at least an inverse polynomial, Theorem 7.1.5 yields stronger average-case hardness result against $\mathcal{C}$.

The proof of Theorem 7.1.5 uses a diagonalization trick similar to the one used for obtaining lower bounds from online learning algorithms in order to construct a family of functions $\mathbb{G}$ such that for every $c \in C$ there is some $g \in \mathbb{G}$ that weakly *approximates* c. We can then apply recent work due to Chattopadhyay et al. [46] relating explicit low-discrepancy colorings to hard on average functions to find an explicit function f that has low correlation with every function in $\mathbb{G}$. This will be the desired hard on average function.

For a subtle technical reason, we need additional assumptions to obtain results for the full SQ model of learning (see Section 7.5.3). We leave getting rid of these assumptions as an interesting open problem, and discuss the difficulty in more detail in Section 7.6.

7.2 Preliminaries and notation

A Boolean function (concept) maps $\{-1,1\}^n \to \{-1,1\}$. A family of Boolean functions $f = \{f_n\}_{n\in\mathbb{N}}$, where $f_n : \{-1,1\}^n \to \{-1,1\}$, naturally corresponds to the language $L_f = \{x \in \{-1,1\}^n \mid f(x) = -1\}$. We use $\mathcal{U}$ (or $\mathcal{U}_n$) to denote the uniform distribution on $\{-1,1\}^n$.

We will use $\mathcal{C} = \cup_{n\in\mathbb{N}} \mathcal{C}_n$ to denote a representation class of Boolean functions, such as DNFs, Boolean circuits, depth-two threshold circuits, etc. The size of $c \in \mathcal{C}$ in its representation will be denoted by $\mathsf{size}(c)$. For concreteness, $\mathsf{size}(c)$ can be assumed to be the number of bits required to write down the representation of c. We require the representation be such that the value at any input of any function c can be computed in deterministic time polynomial in n and the size of the representation. We will use T for denoting time bounds, and s for denoting sizes of representations, both of which we assume to be constructive and non-decreasing without explicit notice.

We now set up some notation to talk about languages and representation classes.

Definition 7.2.1 (Languages and Representation Classes)**.** *For any language $L \subseteq \{-1,1\}^*$, we denote the restriction of L to strings of length n by L_n. For any size function $s : \mathbb{N} \to \mathbb{N}$ and representation class $\mathcal{C}$,*

$$\mathcal{C}^s = \{L \subseteq \{-1,1\}^* \mid \forall n\ \exists c \in \mathcal{C}_n \text{ with } \mathsf{size}(c) \leq s \text{ such that } x \in L \Leftrightarrow c(x) = -1\}.$$

Let $\mathsf{P/poly}[\mathcal{C}] = \cup_{c>0}\mathcal{C}^{n^c}$. *When* $\mathcal{C}$ *is the class of circuits of* AND, OR *and* NOT *gates, we denote* $\mathcal{C}^s$ *by* $\mathsf{SIZE}(s)$ *and* $\mathsf{P/poly}[\mathcal{C}]$ *by just* $\mathsf{P/poly}$.

As such each one of our results will hold for sufficiently large n and we will skip noting this explicitly in the interest of clarity. If we need to stress that we are dealing with functions in $\mathcal{C}$ of n dimensions, we will make this explicit by writing $\mathcal{C}_n^s$ for the class $\mathcal{C}^s$.

To denote that an algorithm has oracle access to a function family f, we write $\mathcal{A}^f$. Equivalently, if we see the oracle as a language L, we write $\mathcal{A}^L$.

We now define the various learning models we will deal with in this chapter. We do not require any of our learning algorithms to be *proper*, that is, the hypothesis output by the algorithms need not be from the representation classes they learn.

7.2.1 Online Mistake Bound Learning

In the mistake-bounded model of learning, a concept c from some class $\mathcal{C}$ is fixed, and a learner is presented with an arbitrary sequence of examples. After receiving each example x_i, the learner must output its prediction for $c(x_i)$. The learner succeeds with mistake bound M if for any sequence of examples, the learner makes at most M mistakes. Formally:

Definition 7.2.2 (Mistake Bound Learning)**.** *Let* $\mathcal{C}$ *be any class of Boolean functions over an arbitrary domain* X. *A mistake bound learning algorithm* $\mathcal{A}$ *for* $\mathcal{C}$ *proceeds in rounds. Let* $c \in \mathcal{C}$ *be the target function. In round* $i \geq 1$, *algorithm* $\mathcal{A}$*:*

(1) *is presented with an example point* $x_i \in X$, *and outputs a label* $\mathcal{A}(x_i)$.

(2) *is provided (by the target function oracle) with the correct label* $c(x_i)$.

(3) *runs an update procedure.*

$\mathcal{A}$ *learns* $\mathcal{C}^s$ *with mistake bound* $M(n,s)$ *and time* $T(n,s)$, *if for any* $c \in \mathcal{C}^s$ *and any (possibly infinite) sequence of examples from* X, $\mathcal{A}$ *makes at most* $M(n,s)$ *mistakes while outputting the labels, and the update procedure runs in time* $T(n,s)$.

7.2.2 Angluin's model of Exact Learning

Angluin's model of exact learning [17] provides the learner with more powerful access to the target function than the Online Mistake Bound Learning Model. It can be easily shown that any mistake bound algorithm can be translated into an exact learner in Angluin's model while preserving efficiency.

Let $c \in \mathcal{C}^s$ be a target function. In this model, the learning algorithm is allowed to ask two kinds of queries about c to the target function oracle:

- *Membership Queries*: the learner presents a point $x \in \{-1,1\}^n$ and the target function oracle replies with $c(x)$.
- *Equivalence Queries*: the learner presents a Boolean function $\tilde{h} : \{-1,1\}^n \to \{-1,1\}$ to the oracle (represented as a circuit). If $\tilde{h} = c$, the oracle responds with "yes". Otherwise, the oracle responds with "not equivalent", and provides a counter example $x \in \{-1,1\}^n$ such that $\tilde{h}(x) \neq c(x)$.

We can now define an exact learning algorithm for a class of Boolean functions $\mathcal{C}^s$.

Definition 7.2.3 (Exact Learning in Angluin's Model)**.** *A deterministic algorithm $\mathcal{A}$* exact learns *a representation class of Boolean functions $\mathcal{C}^s$ in time $T(n,s)$ and queries $Q(n,s)$ if for any target function $c \in \mathcal{C}^s$, $\mathcal{A}$ makes at most $Q(n,s)$ membership and equivalence queries to the oracle for c and outputs a hypothesis $h : \{-1,1\}^n \to \{-1,1\}$ such that $h(x) = c(x)$ for all $x \in \{-1,1\}^n$ in time $T = T(n,s)$. Further, we assume that any equivalence query, $\tilde{h}$, is computable in time $O(T)$ on any input.*

7.2.3 PAC Learning

In the most common PAC learning framework, there is an unknown concept $c \in \mathcal{C}_n$ to be learned, and the learning algorithm receives random examples labelled consistently with c according to some fixed but unknown distribution $\mathcal{D}$ over $\{-1,1\}^n$. Here we concentrate on the stronger model in which the learner can ask membership queries (present any point x and obtain the value of target function $c(x)$) about the unknown concept, and only needs to

learn under the uniform distribution. In other words, we prove circuit lower bounds from a weaker assumption, namely, the existence of learning algorithms in a more powerful model.

Definition 7.2.4. *Let $\mathcal{C}$ be any class of Boolean functions. An algorithm $\mathcal{A}$* PAC*-learns $\mathcal{C}$ if for every $c \in \mathcal{C}$ and for any $\epsilon, \delta > 0$, given membership query access to c, algorithm $\mathcal{A}$ outputs with probability at least $1-\delta$ over its internal randomness, a hypothesis h such that $\Pr_{x \sim \mathcal{U}_n}[c(x) \neq h(x)] \leq \epsilon$.*

We measure the running time of $\mathcal{A}$ as a function $T = T(n, 1/\epsilon, 1/\delta, \mathsf{size}(c))$, and require that h itself can be evaluated in time at most T. We say that $\mathcal{A}$ is efficient if T is bounded above by a fixed polynomial in its parameters.

7.2.4 Statistical Query Learning

Statistical query learning, defined by Kearns et al. [118] is a natural variant of PAC-learning when the underlying data is noisy. We start with the definition of Statistical Queries (SQs).

Definition 7.2.5 (Statistical Query Oracles and SQs)**.** *Let $\mathcal{C}$ be a concept class on $\{-1,1\}^n$. For any $c \in \mathcal{C}$, a statistical query oracle for c of tolerance $\tau > 0$, $\mathsf{STAT}(c, \tau)$, takes input any representation of a bounded function $\psi : \{-1,1\}^n \times \{-1,1\} \to [-1,1]$ and returns $v \in [-1,1]$ such that $|\operatorname{E}_{x \sim \mathcal{U}}[\psi(x, c(x))] - v| \leq \tau$. A query function ψ is said to be target independent if for every $x \in \{-1,1\}^n$ and $y \in \{-1,1\}$, $\psi(x,y) = \psi(x,-y)$, that is ψ doesn't depend on the target function c.*

The Correlational Statistical Query (CSQ) Oracle is a less powerful version of the SQ oracle which answers only correlational queries.

Definition 7.2.6 (Correlational Statistical Query Oracle)**.** *Let $\mathcal{C}$ be a concept class on $\{-1,1\}^n$ and $\mathcal{D}$ be any distribution on $\{-1,1\}^n$. For any $c \in \mathcal{C}$, a correlational statistical query oracle for c of tolerance $\tau > 0$, $\mathsf{CSTAT}(c, \tau)$, takes input any representation of a bounded function $\psi : \{-1,1\}^n \to [-1,1]$ and returns $v \in [-1,1]$ such that $|\langle c, \psi \rangle_{\mathcal{D}} - v| \leq \tau$.*

We now define learning from SQs and CSQs. We note that CSQ learning algorithms are equivalent to Valiant's [194] model of Evolvability [64].

Definition 7.2.7 (Correlational Statistical Query Learning)**.** *Let $\mathcal{C}$ be a representation class of Boolean functions on $\{-1,1\}^n$. A (Correlational) Statistical Query learning algorithm $\mathcal{A}$ learns $\mathcal{C}^s$ on the uniform distribution in time $T = T(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ and queries $Q = Q(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ if, for any $c \in \mathcal{C}^s$ and any $\epsilon \geq \tau > 0$, $\mathcal{A}$ makes Q queries to $\mathsf{STAT}(c,\tau)$ ($\mathsf{CSTAT}(c,\tau)$) and uses at most T time units to return a hypothesis h such that*

$$\Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)] \leq \epsilon.$$

$\mathcal{A}$ is said to be efficient if both T and Q depend polynomially on $n, \frac{1}{\epsilon}, \frac{1}{\tau}$ and s.

Bshouty and Feldman ([39]) noted that any Statistical Query can be simulated by 2 target independent queries and 2 correlational queries. We include their simple proof for completeness.

Proposition 7.2.8 (Bshouty and Feldman [39])**.** *Any statistical query can be decomposed into two statistical queries that are independent of the target and two correlational queries.*

Proof. Let ψ be a statistical query, and let c be the target function. The result follows immediately from the following equation:

$$\begin{aligned}\mathbb{E}[\psi(x,c(x))] &= \mathbb{E}\left[\psi(x,-1)\frac{1-c(x)}{2} + \psi(x,1)\frac{1+c(x)}{2}\right] \\ &= \frac{1}{2}\mathbb{E}[\psi(x,1)c(x)] - \frac{1}{2}\mathbb{E}[\psi(x,-1)c(x)] + \frac{1}{2}\mathbb{E}[\psi(x,1)] + \frac{1}{2}\mathbb{E}[\psi(x,-1)].\end{aligned}$$

□

Using Proposition 7.2.8, we will assume that any SQ algorithm $\mathcal{A}$ learning $\mathcal{C}^s$ actually makes only target independent and correlational queries. Further, we will assume that each target independent query is a Boolean function specified by a circuit of size poly(s).

7.3 Lower bounds from mistake-bounded and exact learning algorithms

In this section we give a simple and direct method for constructing a hard function given a (deterministic) mistake-bounded or exact learning algorithm. Specifically, we will

show that if $\mathcal{C}^s$ (here $\mathcal{C}^s$ equals all concepts in $\mathcal{C}$ of size at most s) is learnable in the Online Mistake Bound Model [130] (or the Exact Learning Model [17]) with mistake bound (or number of queries for Exact Learning) less than 2^n, then there is a function computable in a uniform time class that is not computable by any function in $\mathcal{C}^s$. As a corollary, we will see that polynomial-time (deterministic) mistake-bounded or exact learning algorithms for even linear-sized circuit classes implies that $\mathsf{P} = \mathsf{BPP}$. Previous work relating learning algorithms to circuit lower bounds obtained only subexponential-time simulations of BPP.

Our proof shows how to use a mistake-bounded or exact learning algorithm to "fool" every circuit from some large class. We do this by iteratively presenting the learning algorithm labeled examples from *different* functions at each iteration (typically a learning algorithm will only succeed if it is presented with labeled examples from a function fixed in advance from within the class). Recall that our goal here is *not* to obtain an accurate hypothesis, but to construct a hard function using the learning algorithm as a (black box) subroutine. The running time of the algorithm for evaluating the hard function on any input is dependent on the time and mistake bound (or queries for exact learning) of the learning algorithm.

7.3.1 Lower bounds from mistake bounds

In this section, we present our "diagonalization" trick and show that the existence of a Mistake-Bounded learning algorithm for a class $\mathcal{C}$ yields an explicit hard function for $\mathcal{C}$:

Theorem 7.3.1 (Mistake Bound Learning yields Lower Bounds)**.** *Let $\mathcal{C}$ be any class of Boolean functions. Suppose there exists an algorithm $\mathcal{A}$ that learns any $c \in \mathcal{C}$ with mistake bound $M = M(n, s)$ and time $T = T(n, s)$, where $s = s(n)$. Then, for any size s such that $M < 2^n$, there exists a function $f \in \mathsf{DTIME}(M \cdot T)$ such that for any $c \in \mathcal{C}^s$, we have*

$$\Pr_{x \sim \mathcal{U}_n}[f(x) \neq c(x)] > \frac{1}{M+1} - \frac{1}{2^n}.$$

Proof. We must define function f and prove that it cannot be computed by any circuit of size s (pseudocode for the hard function f is given in Algorithm 1). To do this, we describe f's behavior on input x.

Algorithm 1 Hard function f that uses mistake-bounded learner $\mathcal{A}$ as a subroutine

Require:

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

Output: A value in $\{-1,1\}$.

1: Set $t = M + 1$, where $M = M(n, s(n))$, and let $\{-1,1\}^n$ be partitioned into sequential blocks $E_1, E_2, \ldots, E_k$ of size t, where $k = \lceil 2^n/t \rceil$ (the last block may contain less than t points). Initialize function ℓ on E_j to the constant -1.
2: Find j such that $x \in E_j$. Let $t' = |E_j|$ (note that $t' = M + 1$ for any $j < k$).
3: Obtain all the points in E_j and order them lexicographically as $\{x_1, x_2, \cdots x_{t'}\}$.
4: Start simulating the learner $\mathcal{A}$ on the sequence of points $\{x_1, x_2, \cdots x_{t'}\}$.
5: **for** $i = 1$ to t' **do**
6: Simulate $\mathcal{A}$ by presenting x_i and obtain prediction: $\mathcal{A}(x_i)$.
7: Tell the learner $\mathcal{A}$ that it made a mistake and report true label of x_i as $\ell(x_i) = -\mathcal{A}(x_i)$.
8: Simulate the update procedure of $\mathcal{A}$.
9: **end for**
10: **return** $\ell(x)$.

Let $\{-1,1\}^n$ be partitioned into consecutive blocks in lexicographic order $E_1, \ldots, E_k$, each of size $t = M+1$ (the last block may have size smaller than t). A function ℓ is initialized to equal -1. On input x, f determines $j \in [k]$ such that $x \in E_j$. It then simulates learner $\mathcal{A}$ for $t' = |E_j|$ iterations by presenting it examples from E_j in lexicographic order. Let $\{x_1, x_2, \ldots, x_t\}$ be the examples in E_j. On the i^{th} iteration, f simulates $\mathcal{A}$ and presents it with example x_i. The learner responds with its prediction, $\mathcal{A}(x_i)$. The function f sets $\ell(x_i) = -\mathcal{A}(x_i)$ and informs the learner that it has made a mistake. The function f then simulates the update procedure of $\mathcal{A}$ by using the "true" label of x_i, namely $\ell(x_i)$. At the end of $|E_j|$ iterations, f halts and outputs $f(x) = \ell(x_i)$. Since $x \in E_j$, f halts in at most t iterations. Clearly f can be computed on any input in time $O(M \cdot T)$.

Assume for a moment that for any $c \in \mathcal{C}^s$, functions f and c differ in at least one point in E_j whenever $|E_j| = M + 1$. Then if $|E_j| = M + 1$ for each $1 \leq j \leq k$, we have that $\Pr_{x \sim \mathcal{U}_n}[h(x) \neq c(x)] = \frac{1}{M+1}$. If, on the other hand, $|E_k| < M + 1$, then $\Pr_{x \sim \mathcal{U}_n}[h(x) \neq c(x)] \geq \frac{1}{M+1} \cdot (1 - \frac{|E_k|}{2^n}) > \frac{1}{M+1} - \frac{1}{2^n}$.

Let $j < k$ so $|E_j| = M + 1$. To see why f and c differ on E_j, observe that if there exists a $c \in \mathcal{C}^s$ consistent with ℓ on all the examples in E_j, then the sequence of examples

$\{x_1, \ldots, x_t\}$ and labels given to $\mathcal{A}$ by f are consistent with c. But we have forced the learner to make exactly $M + 1$ mistakes on this sequence. This is a contradiction to the mistake bound of $\mathcal{A}$. Thus, the labeling given by ℓ for $\{x_1, \ldots, x_t\}$ cannot be consistent with any $c \in \mathcal{C}^s$. □

7.3.2 Exact Learning yields circuit lower bounds

In this section we show that the existence of an algorithm that learns a class $\mathcal{C}$ in Angluin's model of exact learning using less than 2^n membership and equivalence queries implies lower bounds against $\mathcal{C}$. Learnability in mistake-bounded model implies learning in the exact model, thus the results presented here are stronger than those stated in the previous section. On the other hand, we do not obtain an average case hard function as we could from a mistake-bounded algorithm. In the proof, we make use of a similar "diagonalization" trick, but there are a few more complications involved in simulating the equivalence and membership queries.

Theorem 7.3.2 (Exact Learning yields Lower Bounds)**.** *Let $\mathcal{C}$ be a class of Boolean functions. Suppose there exists an exact learning algorithm $\mathcal{A}$ that exact learns any target function $c \in \mathcal{C}$ in time $T = T(n, s)$ and $< 2^n$ equivalence and membership queries. Then there exists a function $f \in \mathsf{DTIME}(T^2)$ such that $f \notin \mathcal{C}^s$.*

Proof. As in the previous section, we describe a procedure to compute f using blackbox access to the exact learning algorithm $\mathcal{A}$. We will show that $f \notin \mathcal{C}^s$ and $f \in \mathsf{DTIME}(T^2)$. Let x be the input to f. Then f simulates the learner $\mathcal{A}$ and must give responses to the membership queries and equivalence queries that $\mathcal{A}$ makes. The function f keeps track of the membership queries made by $\mathcal{A}$ and counterexamples (in response to equivalence queries made by $\mathcal{A}$) in the set S. If $\mathcal{A}$ makes a membership query and asks for the label of w, and $w \notin S$, f replies with -1, adds w to the set S, and defines $\ell(w) = -1$. Otherwise f responds with $\ell(w)$. If $\mathcal{A}$ makes an equivalence query for hypothesis $\tilde{h} : \{-1, 1\}^n \to \{-1, 1\}$, f replies with "not equivalent", returns counterexample y, the lexicographically first string not in S (recall that $Q < 2^n$), and adds y to S. In addition, f sets $\ell(y) = -\tilde{h}(y)$.

If during f's simulation, $\mathcal{A}$ halts and outputs a hypothesis h, then f chooses a string y, the lexicographically smallest string not in S, adds y to S, and sets $\ell(y) = -h(y)$. This

guarantees that ℓ differs from h on at least one point in S. Finally, we describe what f should output on input x. If $x \in S$, output $\ell(x)$. Otherwise, output -1.

We will need the following simple claim:

Claim 7.3.3. *Suppose an exact learner $\mathcal{A}$ for $\mathcal{C}^s$, running in time $T = T(n, s)$ that makes at most $Q = Q(n, s) < 2^n$ queries is provided answers to all its membership and equivalence queries that are consistent with some $c \in \mathcal{C}^s$. Let S be the union of the set of all membership queries made by $\mathcal{A}$ and the set of all counterexamples presented to $\mathcal{A}$. Then, if any $c' \in \mathcal{C}^s$ satisfies $c'(x) = c(x)$ for all $x \in S$ then, $c(x) = c'(x)$ for every $x \in \{-1, 1\}^n$.*

Proof of Claim. Since $\mathcal{A}$ is an exact learner and all the membership and equivalence queries made by it are answered with replies consistent with c, $\mathcal{A}$ must halt in at most T steps after making at most Q queries with a hypothesis h such that $h(x) = c(x)$ for every $x \in \{-1, 1\}^n$. On the other hand, since $c'(x) = c(x)$ for every $x \in S$, the answers for the membership and equivalence queries received by $\mathcal{A}$ are consistent with c' also, and thus, $h(x) = c'(x)$ for every $x \in \{-1, 1\}^n$. □

We will now argue that $f \notin \mathcal{C}^s$. We need the following notation: let $S_{\mathcal{A}}$ be the value of S and $\ell_{\mathcal{A}}$ be the value of ℓ when f stops simulating $\mathcal{A}$. Similarly, let S_f be the value of S and ℓ_f the value of ℓ when f halts (recall that S_f and ℓ_f differ from $S_{\mathcal{A}}$ and $\ell_{\mathcal{A}}$ only if $\mathcal{A}$ returns a hypothesis h before f stops simulating it, in which case $S_{\mathcal{A}} \subset S_f$ and $\ell_{\mathcal{A}}$ and ℓ_f agree on all points in $S_{\mathcal{A}}$).

Suppose that there exists $c \in \mathcal{C}^s$ such that $f(x) = c(x)$ for every $x \in S_{\mathcal{A}}$. In other words, the replies to the queries made by the algorithm $\mathcal{A}$ are consistent with c. In this case, $\mathcal{A}$ must halt and return a hypothesis h in at most T steps. Moreover, since $\mathcal{A}$ is an exact learner, $h = c$.

By Claim 7.3.3, c is the unique function in $\mathcal{C}^s$ that is consistent with f on $S_{\mathcal{A}}$. Thus, if f is computed by some function in $\mathcal{C}^s$, then $f = c$. But notice that, in this case, the procedure for computing f guarantees that there exists a $y \in S_f \setminus S_{\mathcal{A}}$ such that $f(y) \neq h(y)$. This implies that $f \neq c$. Thus, there is no function in $\mathcal{C}^s$ that computes f.

On the other hand if for every $c \in \mathcal{C}^s$, there is some value $x \in S_{\mathcal{A}}$ such that $f(x) \neq c(x)$, then we immediately conclude that f is not computed by any $c \in \mathcal{C}^s$. In either case, we

have proved that $f \notin \mathcal{C}^s$.

The function f can simulate $\mathcal{A}$ in time $O(T)$. Since $\mathcal{A}$ makes at most T equivalence queries, each of which is computable at any point in deterministic time $O(T)$ (Section 7.2.2), $\mathcal{A}$ spends at most $O(T^2)$ time answering equivalence queries. All other computations of f involve searching for strings outside S which takes at most $O(S) = O(T)$ time. Thus we have that f runs in time at most $O(T^2)$. □

As a simple corollary we obtain that efficient exact learnability of $\mathcal{C}$ yields $\mathsf{DTIME}(n^{\omega(1)}) \nsubseteq \mathsf{P/poly}[\mathcal{C}]$. We now apply the theorem above to the special case of $\mathsf{SIZE}(n)$ to compare our results with [70] and [92].

Corollary 7.3.4. *Suppose* $\mathsf{SIZE}(n)$ *is learnable -*

- *by a Mistake-Bounded Algorithm in time and mistake bound polynomial in* n*; or*
- *by an Exact Learning Algorithm in time polynomial in* n*.*

Then, $\mathsf{DTIME}(n^{\omega(1)}) \nsubseteq \mathsf{P/poly}$.

Proof. By a simple padding argument, $\mathsf{P/poly}$ is efficiently learnable in the respective models. Applying Theorems 7.3.1 and 7.3.2 yields the result. □

For a comparison, note that [70] proves that if $\mathsf{P/poly}$ is efficiently exactly learnable in Angluin's model, then $\mathsf{EXP}^{\mathsf{NP}} \nsubseteq \mathsf{P/poly}$, and [92] improves this result to obtain the conclusion that $\mathsf{EXP} \nsubseteq \mathsf{P/poly}$.

7.3.3 Derandomization consequences from Exact Learners

The improvements in our lower bounds allow us to obtain a complete derandomization of BPP from efficient learnability of $\mathsf{P/poly}$.

We will require the following celebrated result of Impagliazzo and Wigderson:

Theorem 7.3.5 (Impagliazzo and Wigderson [101])**.** *If there exists* $L \in \mathsf{DTIME}(2^{O(n)})$ *and* $\delta > 0$ *such that* $L \notin \mathsf{SIZE}(2^{\delta n})$*, then* $\mathsf{P} = \mathsf{BPP}$.

Previous work obtained only subexponential deterministic simulations of BPP given the existence of efficient learning algorithms.

Corollary 7.3.6. *Suppose* $\mathsf{SIZE}(n)$ *is efficiently learnable in Angluin's model of exact learning with membership and equivalence queries. Then* $\mathsf{P} = \mathsf{BPP}$.

We only state the above corollary starting from exact learning algorithms, as mistake-bounded learnability implies exact-learnability.

Proof. We again use a padding argument here, although we have to be a bit more explicit with our parameters. Suppose $\mathsf{SIZE}(n)$ is exactly learnable in time $O(n^k)$. By padding $\mathsf{SIZE}(s)$ is learnable in time $O(s^k)$. Let $s = 2^{\delta n}$, where $\delta = \frac{1}{2k}$. The result now follows easily from Theorem 7.3.2 and Theorem 7.3.5. □

We note that using similar tools from derandomization, we can show that the existence of sub-exponential time mistake-bounded learning algorithms for polynomial size circuits implies subexponential-time derandomization of BPP.

7.4 Lower bounds from PAC learning algorithms

In this section we shift gears and obtain hard functions in PSPACE from PAC learning algorithms. Previous work [70] showed the existence of hard functions in BPEXP. Indeed, here we prove that unless randomness can speed-up arbitrary space-bounded computations, any non-trivial PAC learning algorithm for a circuit class $\mathcal{C}$ yields a hard function in PSPACE against $\mathcal{C}$. We begin with a few important definitions.

Definition 7.4.1 (Downward Self-Reducibility)**.** *We say that a language* L *is* downward-self-reducible *if there is a deterministic polynomial time algorithm* A *such that for all* $x \in \{-1,1\}^n$, $A^{L_{n-1}}(x) = L(x)$. *In other words,* A *efficiently computes* $L(x)$ *on any input* x *of size* n *when given oracle access to a procedure that computes* L *on inputs of size* $n-1$.

Definition 7.4.2 (Self-Correctibility)**.** *We say that a language* L *is* $\alpha(n)$-self-correctible *if there is a probabilistic polynomial time algorithm* A *such that, for any Boolean function* $c : \{-1,1\}^n \to \{-1,1\}$ *that disagrees with* L_n *on at most an* $\alpha(n)$*-fraction of the inputs of size* n, *we have* $\Pr[A^c(x) = L(x)] \geq 2/3$ *for any* $x \in \{-1,1\}^n$.

Using an appropriate arithmetization of quantified Boolean formulas, Trevisan and Vadhan [187] proved that there exists a PSPACE-complete language that is both downward-self-reducible and self-correctible. Actually, by employing better self-correction techniques introduced by Gemmel and Sudan [76] and a standard composition with the Hadamard error-correcting code, it follows from their construction that [189]:

Proposition 7.4.3. *There exists a* PSPACE*-complete language* L_{PSPACE} *that is both downward-self-reducible and* α*-self-correctible, where* $\alpha = 1/100$.

Finally, for any language $\mathcal{O}$, we denote by $\mathsf{BPTIME}(T(n))^{\mathcal{O}}$ the class of languages that can be computed by probabilistic algorithms that have oracle access to $\mathcal{O}$ and run in time $O(T(n))$.

Theorem 7.4.4 (PAC Learning yields Lower Bounds)**.** *Let* $\mathcal{C}$ *be any concept class and suppose that there exists an algorithm that* PAC *learns any* $c \in \mathcal{C}^s$ *under the uniform distribution using membership queries when given access to an oracle* $\mathcal{O}$[1] *in time* $T(n, 1/\epsilon, \log 1/\delta, s)$. *Let* $L^\star$ *be a language that is both downward-self-reducible and* $\alpha(n)$*-self-correctible. Then, at least one of the following conditions hold:*

(*i*) $L^\star \notin \mathcal{C}^s$*; or*

(*ii*) $L^\star \in \mathsf{BPTIME}(\mathsf{poly}(T(n, 1/\alpha(n), \log n, s)))^{\mathcal{O}}$.

The proof of this result follows the same high-level approach employed by Fortnow and Klivans [70], which we sketch next. Suppose for simplicity that we have an efficient PAC learning algorithm for $\mathcal{C}$ that does not depend on any oracle $\mathcal{O}$, and that $L^\star \in \mathsf{P/poly}[\mathcal{C}]$ (otherwise there is nothing to prove). Note that in order to prove Theorem 7.4.4, it is enough to show that $L^\star \in \mathsf{BPP}$. This can be obtained by combining the learning algorithm for $\mathcal{C}$ with the downward-self-reducibility and self-correctibility of $L^\star$. Roughly speaking, we "learn" how to compute $L^\star$ on all inputs of size at most n starting from inputs of constant size, which can be easily computed by a truth-table. Assuming that we know how to compute $L^\star_k$ with high probability on every input, we can compute $L^\star_{k+1}$ with high probability as follows.

[1] We stress that the learner can ask both membership queries about the unknown concept and queries to oracle $\mathcal{O}$.

Simulate the learning algorithm with unknown concept $c = L^\star_{k+1}$. Answer membership queries using downward-self-reducibility and the procedure for $L^\star_k$ obtained by induction. The learner outputs a hypothesis h for $c = L^\star_{k+1}$ that is close to $L^\star_{k+1}$. Now use the self-correctibility of $L^\star$ together with c to obtain an algorithm that computes $L^\star_{k+1}$ on every input with high probability. Observe that each stage can be computed efficiently from our assumptions. After n stages, we obtain a randomized algorithm that computes $L^\star$ on any input of size n, which completes the proof that $L^\star \in \mathsf{BPP}$. For completeness, we present the full proof of Theorem 7.4.4 in Section 7.7.

The next corollary is immediate by taking $\mathcal{O}$ to be the empty language in the statement of Theorem 7.4.4.

Corollary 7.4.5. *Let $\mathcal{C}$ be any concept class and suppose that there exists an algorithm that* PAC *learns any $c \in \mathcal{C}^s$ under the uniform distribution using membership queries in time $T(n, 1/\epsilon, \log 1/\delta, s)$. Also, let L_{PSPACE} be the* PSPACE*-complete language given by Proposition 7.4.3. Then, at least one of the following conditions hold:*

(i) $L_{\mathsf{PSPACE}} \notin \mathcal{C}^s$*; or*

(ii) $L_{\mathsf{PSPACE}} \in \mathsf{BPTIME}(\mathsf{poly}(T(n, O(1), \log n, s)))$.

For instance, for efficient PAC learning algorithms we have the following consequence:

Corollary 7.4.6. *Let $\mathcal{C}$ be any concept class and suppose that there exists a polynomial-time algorithm that* PAC *learns $\mathcal{C}$ under the uniform distribution using membership queries. Then, at least one of the following conditions hold:*

(i) $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}[\mathcal{C}]$*; or*

(ii) $\mathsf{PSPACE} \subseteq \mathsf{BPP}$.

Corollary 7.4.6 implies the original result of Fortnow and Klivans: if $\mathsf{PSPACE} \subseteq \mathsf{BPP}$, a simple padding argument gives $\mathsf{EXPSPACE} \subseteq \mathsf{BPEXP}$, and it is not hard to prove by diagonalization that EXPSPACE requires circuits of size $\Omega(2^n/n)$. Thus, under efficient PAC learnability of $\mathcal{C}$, it follows that either $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}[\mathcal{C}]$ or BPEXP requires circuits of size $\Omega(2^n/n)$. In particular, this implies that $\mathsf{BPEXP} \not\subseteq \mathsf{P/poly}[\mathcal{C}]$.

Note that the second condition in the conclusion above does not depend on the class $\mathcal{C}$ that appears in the hypothesis. We observe next that removing this "or" condition from the conclusion of Corollary 7.4.6 would give us an unconditional proof that $\mathsf{PSPACE} \neq \mathsf{BPP}$. To see this, suppose the following result is valid:

$$\text{If } \mathcal{C} \text{ is PAC-learnable in polynomial-time then } \mathsf{PSPACE} \not\subseteq \mathsf{P/poly}[\mathcal{C}] \ (\star)$$

Let $\mathcal{C}$ be the class of Boolean circuits, i.e., $\mathsf{P/poly}[\mathcal{C}] = \mathsf{P/poly}$. Let $\mathsf{P/poly\text{-}PAC\text{-}learnable}$ denote that $\mathcal{C}$ is PAC learnable in polynomial time. We prove that both $\mathsf{P/poly\text{-}PAC\text{-}learnable}$ and its negation $\neg\mathsf{P/poly\text{-}PAC\text{-}learnable}$ imply $\mathsf{BPP} \neq \mathsf{PSPACE}$. First, if $\mathsf{P/poly\text{-}PAC\text{-}learnable}$ then it follows from $(\star)$ that $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$. Since $\mathsf{BPP} \subseteq \mathsf{P/poly}$ (Adleman [4]), this implies $\mathsf{BPP} \neq \mathsf{PSPACE}$. On the other hand, suppose we have $\neg\mathsf{P/poly\text{-}PAC\text{-}learnable}$. To show that that $\mathsf{BPP} \neq \mathsf{PSPACE}$, it is sufficient to prove that if $\mathsf{BPP} = \mathsf{PSPACE}$ then $\mathcal{C}$ is efficiently PAC-learnable. Using $\mathsf{PSPACE} \subseteq \mathsf{BPP}$, we can find a efficiently find a hypothesis consistent with the labeled examples with high probability and a well known result (Occam's Razor, see Proposition 7.4.7 below) now implies PAC-learning.

7.4.1 A new way to prove Karp-Lipton collapse theorems

In this section we show that Proposition 7.4.7 (Occam's Razor) together with Theorem 7.4.4 (PAC learning yields circuit lower bound) can be used to prove Karp-Lipton style collapse theorems (Karp and Lipton [115]). Recall that these theorems state that if some circuit lower bound does not hold then there is a unexpected collapse involving uniform complexity classes. These results are usually stated with respect to $\mathsf{P/poly}$. The most famous Karp-Lipton Theorem says that if $\mathsf{NP} \subseteq \mathsf{P/poly}$ then $\mathsf{PH} = \Sigma_2^p$, i.e., the polynomial time hierarchy collapses to its second level. Similar theorems are known for different complexity classes. To prove more refined results, we use $\mathsf{SIZE}(l(n))$ to denote the class of languages with circuits of size $O(l(n))$. For concreteness, we give a proof for PSPACE. However, it is clear that the same argument works for any complexity class containing complete problems that are both downward-self-reducible and self-correctible, such as $\#\mathsf{P}$.

We start by stating the Occam's Razor technique [32].

Proposition 7.4.7 (Occam's Razor Principle)**.** *Let $\mathcal{C}$ be any representation class and $s : \mathbb{N} \to \mathbb{N}$ be an arbitrary constructive function. Suppose there exists an algorithm B that, given any set of $m \geq \frac{1}{\epsilon}\left(s + \log\frac{1}{\delta}\right)$ uniformly distributed random examples labelled according to some unknown concept $c \in \mathcal{C}_n^s$, outputs a hypothesis $h \in \mathcal{C}^s$ that is consistent with this set of examples. Then B is a* PAC *learning algorithm for $\mathcal{C}$. In other words, the hypothesis h outputted by B is ϵ-close to c with probability at least $1 - \delta$.*

Proposition 7.4.8. *Let $\mathcal{C}$ be an arbitrary concept class and $s' : \mathbb{N} \to \mathbb{N}$ be any constructive function with $s'(n) \geq n$. If $L_{\mathsf{PSPACE}} \in \mathcal{C}^{s'(n)}$ then $L_{\mathsf{PSPACE}} \in \mathsf{BPTIME}(\mathsf{poly}(s'(n)))^{\mathsf{NP}}$.*

Proof. For any concept class $\mathcal{C}$, there exists an algorithm $\mathcal{B}$ that uses an NP oracle and is able to learn any concept $c \in \mathcal{C}$ in time $T(n, 1/\epsilon, \log 1/\delta, \mathsf{size}(c)) = \mathsf{poly}(n, 1/\epsilon, \log 1/\delta, \mathsf{size}(c))$. This algorithm simply draws $m = \frac{1}{\epsilon}\left(\mathsf{size}(c) + \log\frac{1}{\delta}\right)$ random examples labelled according to c and uses its NP oracle together with a standard search to decision reduction to find a hypothesis $h \in \mathcal{C}^{\mathsf{size}(c)}$ that is consistent with all examples. By Occam's Razor (Proposition 7.4.7), $\mathcal{B}$ is a PAC learning algorithm for $\mathcal{C}$.

Let $L^\star = L_{\mathsf{PSPACE}}$, $\mathcal{O} = \mathsf{NP}$, and $s = s'(n)$ in the statement of Theorem 7.4.4. It follows that either $L_{\mathsf{PSPACE}} \notin \mathcal{C}^{s'(n)}$ or $L_{\mathsf{PSPACE}} \in \mathsf{BPTIME}(\mathsf{poly}(T(n, O(1), \log n, s')))^{\mathsf{NP}} = \mathsf{BPTIME}(\mathsf{poly}(s'(n)))^{\mathsf{NP}}$. The result then follows from the assumption that $L_{\mathsf{PSPACE}} \in \mathcal{C}^{s'(n)}$. □

Corollary 7.4.9. *If $L_{\mathsf{PSPACE}} \in \mathsf{SIZE}(l(n))$ then $L_{\mathsf{PSPACE}} \in \mathsf{BPTIME}(\mathsf{poly}(l(n)))^{\mathsf{NP}}$.*

Corollary 7.4.10. *If $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ then $\mathsf{PSPACE} \subseteq \mathsf{BPP}^{\mathsf{NP}}$.*

We remark that this is not a new result. For instance, it is known that if $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ then $\mathsf{PSPACE} \subseteq \mathsf{MA}$ (Babai, Fortnow, and Lund [22]), and also that $\mathsf{MA} \subseteq \mathsf{ZPP}^{\mathsf{NP}} \subseteq \mathsf{BPP}^{\mathsf{NP}}$ (Goldreich and Zuckerman [81]).

Following the terminology of Trevisan and Vadhan [187], one may interpret our results as a new way to prove "super Karp-Lipton" theorems for PSPACE. For instance, if there exists a polynomial-time learning algorithm for TC^0_2, it follows that $\mathsf{PSPACE} \subseteq \mathsf{TC}^0_2$ implies $\mathsf{PSPACE} = \mathsf{BPP}$.

7.5 Lower bounds from SQ and CSQ learning algorithms

In this section we show that efficient SQ learning algorithms for a class $\mathcal{C}$ of circuits yield circuit lower bounds. We will first show that efficient CSQ algorithms yield explicit average case hard functions and then go on to obtain a non-constructive lower bound from an SQ learning algorithm.

7.5.1 Preliminaries

Definition 7.5.1 (Inner Product)**.** *For any functions f, g mapping $\{-1,1\}^n$ into $\{-1,1\}$, we denote the inner product of f and g with respect to $\mathcal{U}_n$ by $\langle f, g\rangle$. The inner product with respect to the uniform distribution on $X \subseteq \{-1,1\}^n$, $\mathcal{U}_X$, is denoted by $\langle f, g\rangle_X$.*

Definition 7.5.2 (Hamming Distance)**.** *For any two Boolean functions f, g mapping $\{-1,1\}^n$ into $\{-1,1\}$, the hamming distance between f and g denoted by $dist(f,g) = \frac{1}{2^n}|\{x \in \{-1,1\}^n \mid f(x) \neq g(x)\}|$. Observe that $dist(f,g) = \frac{1}{2}(1 - |\langle f, g\rangle|)$. Hamming distance is a metric on the space of Boolean functions on $\{-1,1\}^n$.*

We will now define discrepancy of a bounded function class [47, 136, 149]. The definition we present here (and used in [46]) is a natural generalization of the standard definition of discrepancy to classes of bounded functions.

Definition 7.5.3 (Discrepancy of a Class of Bounded Functions)**.** *Let $\mathcal{C}$ be a class of bounded functions mapping a finite set X into $[-1,1]$ and let $\chi : X \to \{-1,1\}$ be a coloring of X. The discrepancy of χ with respect to a function $c \in \mathcal{C}$ is defined as $\chi(c) = \sum_{x:c(x)\geq 0} \chi(x) \cdot c(x)$. The discrepancy of χ with respect to the class $\mathcal{C}$ on X is defined as $disc[X, \mathcal{C}](\chi) = \max_{c\in\mathcal{C}} |\chi(c)|$.*

A uniformly random coloring is, not surprisingly, a low discrepancy coloring. The proof is a direct application of the Chernoff-Hoeffding Bounds. Further, this procedure to construct a low-discrepancy coloring can be derandomized [176].

Lemma 7.5.4 (Deterministic Construction of Low Discrepancy Coloring [176])**.** *Let $\mathcal{C}$ be a class of bounded functions on X with $|\mathcal{C}| = m$. There exists a deterministic algorithm run-*

ning in time $\mathrm{poly}(m, |X|)$ *that produces a coloring with discrepancy at most* $\sqrt{4|X| \log 4m}$ *for* $\mathcal{C}$.

There is a simple connection between a low discrepancy coloring χ for $\mathcal{C}$ on X and average case hardness of χ for $\mathcal{C}$ observed in [46].

Proposition 7.5.5 (Low Discrepancy ⇒ Average Case Hard Function)**.** *Let* $\mathcal{C}$ *be a class of bounded functions mapping* X *into* $[-1, 1]$. *Let* $-C = \{-c : c \in C\}$ *denote the class of all negated functions from* $\mathcal{C}$. *If* $\chi : X \to \{-1, 1\}$ *is a coloring of* X *with discrepancy at most* $\epsilon|X|$ *with respect to* $\mathcal{C} \cup -\mathcal{C}$ *then* $|\langle \chi, c \rangle_{\mathcal{U}_X}| \leq 2\epsilon$ *for each* $c \in \mathcal{C}$ *on* X.

Proof. Let $c \in \mathcal{C}$. Since χ has discrepancy at most $\epsilon|X|$ with respect to c and $-c$, we have:

$$\Big| \sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) \cdot c(x) \Big| \leq \epsilon|X| \quad \text{and} \quad \Big| \sum_{\substack{x \in X \\ c(x) \leq 0}} \chi(x) \cdot c(x) \Big| \leq \epsilon|X|.$$

Thus,

$$\begin{aligned} |\langle \chi, c \rangle_{\mathcal{U}_X}| &= \Big| \mathrm{E}_{x \sim \mathcal{U}_X}[\chi(x) \cdot c(x)] \Big| \\ &\leq \frac{1}{|X|} \cdot \Big(\Big| \sum_{\substack{x \in X \\ c(x) \geq 0}} \chi(x) \cdot c(x) \Big| + \Big| \sum_{\substack{x \in X \\ c(x) \leq 0}} \chi(x) \cdot c(x) \Big| \Big) \\ &\leq \frac{1}{|X|} \cdot \Big(\epsilon|X| + \epsilon|X| \Big) \\ &= 2\epsilon. \end{aligned}$$

□

7.5.2 CSQ Learning yields circuit lower bounds

To show that CSQ learning algorithms yield circuit lower bounds, we use a learning algorithm $\mathcal{A}$ for $\mathcal{C}$, to construct a small set of functions $\mathbb{G}$ such that each function in $\mathcal{C}$ is non-trivially correlated with some function in $\mathbb{G}$. This construction is well-known, and has been employed in other contexts [64].

Lemma 7.5.6 (Small Weakly Correlating set from CSQ Algorithm)**.** *Let* $\mathcal{C}$ *be a representation class of Boolean functions on* $\{-1, 1\}^n$. *Suppose for some* ϵ, τ *such that* $\frac{1}{2} > \epsilon \geq \tau > 0$ *and* $\tau \leq 1 - 2\epsilon$, $\mathcal{C}^s$ *is learnable on the uniform distribution in the* CSQ *model by an algorithm*

$\mathcal{A}$ running in time $T = T(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ while making at most $Q = Q(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ correlational queries of tolerance τ. Then, there exists a set $\mathbb{G}$ of at most $Q + 1$ functions mapping $\{-1, 1\}^n$ into $[-1, 1]$ such that, for every $c \in \mathcal{C}^s$, there exist a $g \in \mathbb{G}$ such that $|\langle g, c\rangle| \geq \tau$. Moreover, such a set $\mathbb{G}$ can be recovered by an algorithm running in deterministic time T.

Proof. We will simulate the CSQ oracle for $\mathcal{A}$ and simulate the learning algorithm to construct the set of functions $\mathbb{G}$.

Simulate the CSQ algorithm $\mathcal{A}$ for $\mathcal{C}^s$. Each time the algorithm makes a correlational query to the CSQ oracle, return 0. Stop the simulation if $\mathcal{A}$ runs for T steps or makes Q queries. Let $g_1, g_2, \cdots, g_k$ be the queries made by the algorithm after we stop the simulation. Then, $k \leq Q$. If $\mathcal{A}$ doesn't return any hypothesis, there must be no function in $\mathcal{C}^s$ consistent with our answers for the CSQs, which immediately yields that for every $c \in \mathcal{C}^s$, there exists a g_i for $i \in [k]$ such that $|\langle c, g_i\rangle| > \tau$. If $\mathcal{A}$ returns a hypothesis, call it h, and let $\mathbb{G} = \{g_i \mid 1 \leq i \leq k\} \cup \{h\}$.

We now verify that $\mathbb{G}$ satisfies the required conditions stated in the theorem. Let $c \in \mathcal{C}^s$. One of the following two conditions has to be true:

1. $|\langle c, g_i\rangle| \leq \tau$ for each $1 \leq i \leq k$.

   In this case observe that the answers returned to the algorithm while simulating the CSQ oracle are consistent with the target function c within the tolerance bound of τ. Thus, $\Pr_{x\sim\mathcal{U}}[h(x) \neq c(x)] \leq \epsilon$, which gives $|\langle c, h\rangle| \geq 1 - 2\epsilon \geq \tau$.

2. There exists a j with $1 \leq j \leq k$ such that $|\langle g_j, c\rangle| \geq \tau$.

   In this case we are immediately done, since $g_j \in \mathbb{G}$.

□

We now show that CSQ learning algorithms yield circuit lower bounds.

Theorem 7.5.7 (CSQ Learning yields circuit lower bounds)**.** *Let $\mathcal{C}$ be a representation class of Boolean functions on $\{-1, 1\}^n$. Let ϵ, τ be any parameters satisfying $\epsilon < \frac{1}{2}$ and $2^{-o(n)} \leq \tau \leq \min\{\epsilon, 1 - 2\epsilon\}$. Suppose there exists an algorithm $\mathcal{A}$ that runs in time $T = T(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ that learns $\mathcal{C}^s$ on the uniform distribution in the CSQ model to accuracy $1 - \epsilon$ by at most $Q = Q(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s) \leq 2^n$ queries, each of tolerance τ. Then, there exists a Boolean function (family) $f \in \mathsf{DTIME}(T + \mathrm{poly}(Q, \frac{1}{\tau}))$ such that for every $c \in \mathcal{C}^s$, $\Pr_{x\sim\mathcal{U}}[f(x) \neq c(x)] \geq \frac{\tau}{4}$.*

The intuitive idea is that we can use Lemma 7.5.6 to construct the set $\mathbb{G}$ of size $\leq Q+1$. Running deterministic discrepancy minimization algorithm (Lemma 7.5.4) on $\mathbb{G} \cup -\mathbb{G}$ yields a function f that has low correlation with every function in $\mathbb{G}$ (using Proposition 7.5.5). The fact that every function in $\mathcal{C}^s$ is non-trivially correlated with some function in $\mathbb{G}$ is then invoked to argue that f should be far from $\mathcal{C}^s$.

Remark 5. *The theorem holds for any $\epsilon < 1/2$ and thus even a weak learning algorithm for $\mathcal{C}^s$ that uses only* CSQ*s yields lower bounds against $\mathcal{C}^s$.*

Algorithm 2 Hard Function f that uses CSQ learner $\mathcal{A}$ as a subroutine

Require:

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

Output: A value in $\{-1,1\}$.

1: Use learner $\mathcal{A}$ to obtain the set $\mathbb{G}$ of size at most $Q+1$ of weakly correlating functions for $\mathcal{C}^s$.
2: Let $\{-1,1\}^n$ be partitioned into consecutive blocks in lexicographic order $E_1, E_2, \cdots, E_k$ each of size t (the last block may be of smaller size).
3: Determine j such that $x \in E_j$. Recover all the points in E_j.
4: Run deterministic discrepancy minimization on the class $\mathbb{G} \cup -\mathbb{G}$ and domain E_j to obtain a function $f_j : E_j \to \{-1,1\}$.
5: Return $f_j(x)$.

Proof. We need to describe a procedure to compute f using blackbox access to the CSQ learning algorithm $\mathcal{A}$. We will show that f is far from $\mathcal{C}^s$ and $f \in \mathsf{DTIME}(T + \mathrm{poly}(Q))$. Let x be the input to f. First, construct $\mathbb{G}$, the set of weakly correlating functions by simulating the learning algorithm for $\mathcal{C}^s$ using Lemma 7.5.6. Notice that since the CSQ algorithm doesn't use any randomness of its own, the procedure produces a fixed set $\mathbb{G}$ in any run of the algorithm. Let $\{-1,1\}^n$ into consecutive blocks $E_1, E_2, \cdots E_k$ each of size $t = \lceil \frac{64 \log 2|\mathbb{G}|}{\tau^2} \rceil \leq \lceil \frac{64 \log 4Q}{\tau^2} \rceil$ (the last block E_k may be smaller). f first finds out j such that $x \in E_j$. It then runs the deterministic discrepancy minimization algorithm (Lemma 7.5.4) on the class $\mathbb{G} \cup -\mathbb{G}$ and domain E_j. Suppose $f_j : E_j \to \{-1,1\}^n$ is the function returned by the algorithm. f outputs $f_j(x)$. We refer the reader to Algorithm 2 for the pseudo-code of this routine.

By using Proposition 7.5.5, we observe that for every $g \in \mathbb{G}$, and for every $j < k$,

$$|\langle f_j, g\rangle_{E_j}| \;\leq\; 2\sqrt{\frac{4\log 4Q}{t}} \;\leq\; \tau/4.$$

Fix any $j \in [k]$. Notice that for any $x \in E_j$, the algorithm runs the discrepancy minimization on the same class $\mathbb{G} \cup -\mathbb{G}$ and on the same domain E_j, thus constructing the same function f_j each time. Thus, for each $x \in \{-1,1\}^n$, $f(x) = f_j(x)$ whenever $x \in E_j$. Therefore, for each g,

$$|\langle f, g\rangle| \;=\; \Big|\sum_{j=1}^{k} \frac{|E_j|}{2^n}\langle f_j, g\rangle_{E_j}\Big| \;\leq\; \frac{\tau}{4}\cdot\Big(\sum_{j=1}^{k-1}\frac{|E_j|}{2^n}\Big) + 1\cdot\frac{t}{2^n} \;<\; \frac{\tau}{4} + \frac{1}{2^n}\cdot\Big\lceil\frac{64\log 4Q}{\tau^2}\Big\rceil \;\leq\; \tau/2,$$

using our bounds on Q and τ from the statement of the theorem.

To show the average case hardness of f for $\mathcal{C}^s$, fix any $c \in \mathcal{C}^s$. Let $g_c \in \mathbb{G}$ such that $|\langle c, g_c\rangle| \geq \tau$, but $|\langle f, g_c\rangle| \leq \frac{\tau}{2}$. Since by changing a single coordinate in the 2^n-dimensional vector representing function c we can only change the value of $\langle c, g_c\rangle$ by $\pm 2/2^n$, it must be the case that c and f differ in at least a $\tau/4$ fraction of the inputs. In other words, $\Pr_{x\sim\mathcal{U}}[f(x) \neq c(x)] \geq \frac{\tau}{4}$, as desired. Finally, observe that the value of f at any $x \in \{-1,1\}^n$ can be evaluated in deterministic time $\mathrm{poly}(|\mathbb{G}|, \frac{1}{\tau}) = \mathrm{poly}(|Q|, \frac{1}{\tau})$. This completes the proof. □

Note that since the class of all parity functions requires $2^{\Omega(n)}$ SQs to be learned [118], we immediately obtain that if $\mathcal{C}$ is efficiently SQ learnable then $\mathcal{C}$ cannot compute some parity function. Thus any efficient SQ learnability of a class $\mathcal{C}$ immediately yields a worst case lower bound. Such an argument can actually be extended to obtain even a weak average case lower bound. This is based on the characterization of CSQ learning by the SQ dimension studied in [30]. The SQ dimension of a class $\mathcal{C}$ on the uniform distribution is the largest number d such that there exist functions $c_1, c_2, \cdots c_d \in \mathcal{C}$, such that for every $i \neq j$, $|\langle c_i, c_j\rangle| \leq \frac{1}{d^3}$. Kearns et al. characterized the query complexity of the best SQ algorithm that learns $\mathcal{C}$ on $\mathcal{U}$ to be within a polynomial factor of the SQ dimension of $\mathcal{C}$ on $\mathcal{U}$. The following proposition shows that efficient CSQ learnability of $\mathcal{C}$ by CSQs of tolerance τ implies that there is a parity which is $1 - 1/n^{\omega(1)}$-hard for $\mathsf{P/poly}[\mathcal{C}]$.

Compare this to Theorem 7.5.7, which shows from the same assumption that there is a function computable in super-polynomial time that is $\tau/4$-hard for $\mathsf{P/poly}[\mathcal{C}]$.

Lemma 7.5.8. *Suppose $\mathcal{C}^s$, a representation class of Boolean functions of size at most s, is learnable to an accuracy of $1/3$ by CSQs of tolerance τ (lower bounded by an inverse polynomial in n) in time $T(n,s)$ upper bounded by some polynomial in n, s where c is the target function. Then, there exists a parity χ_S, $|S| = O(\frac{\log s}{\log n})$, such that $\Pr_{x\sim\mathcal{U}}[\chi_S(x) \neq c(x)] \geq \frac{1}{n^{|S|}}$ for every $c \in \mathcal{C}^s$. Consequently for every $k = \omega(1)$, there exists a parity that cannot be computed on at least $\frac{1}{n^k}$ fraction of the inputs by any function in $\mathsf{P/poly}[\mathcal{C}]$*

Proof. Suppose $\mathcal{C}^s$ is learnable by a CSQ algorithm to accuracy of $1/3$ in time $T(n,s)$. Then the SQ dimension of $\mathcal{C}^s$ on the uniform distribution is bounded above by $T(n,s)$. For some k, that we will fix later, consider the set of all parities over subsets of at most k variables out of n variables.

Suppose for each $T \subseteq [n]$ such that $|T| \leq k$, there exists a $c_T \in \mathsf{P/poly}[\mathcal{C}]$ such that $|\langle c_T, \chi_T\rangle| > 1 - \frac{1}{n^k}$. Consider $|\langle c_T, c_R\rangle| = 2(1 - dist(c_T, c_R))$ for some T, R such that $|T|, |R| \leq k$. Then, by triangle inequality (for Hamming distance), $dist(\chi_T, \chi_R) \leq dist(\chi_T, c_T) + dist(c_T, c_R) + dist(c_R, \chi_R)$. Using the orthogonality of distinct parities, $dist(c_T, c_R) \geq 1 - 2 \cdot (\frac{1}{2} - \frac{1}{2n^k}) = \frac{1}{n^k}$, yielding $|\langle c_T, c_R\rangle| < 1 - (1 - \frac{1}{n^k}) = \frac{1}{n^k}$. This yields that the set $\{c_T \mid |T| \leq k\}$ forms a set of n^k functions that satisfy $|\langle c_T, c_R\rangle| \leq \frac{1}{n^k}$ for every $T \neq R$ of size k, or that the SQ dimension of $\mathsf{P/poly}[\mathcal{C}]$ is at least n^k.

Now, choose a large enough constant k such that $n^k > T(n,s)$ to obtain a contradiction, yielding that some parity on k variables cannot be correlated with any $c \in \mathcal{C}^s$ by more than $1 - \frac{1}{n^k}$. □

Remark 6. *Observe that from the proof above, one can only obtain the conclusion that some parity differs from every $c \in \mathsf{P/poly}[\mathcal{C}]$ on a negligible fraction (inverse super-polynomial) of inputs.*

7.5.3 SQ Learning yields circuit lower bounds

In this section we prove that it is possible to obtain strong average-case hardness results from the existence of an algorithm that learns using statistical queries. Formally, we will show that if we can learn a class $\mathcal{C}$ in the statistical query model, then either there is an explicit function f that is average-case hard for $\mathcal{C}$, or $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P}$.

We have seen in the proof of Theorem 7.5.7 that a learning algorithm that uses only correlational queries yields an explicit average case hard function. Since the target independent queries do not depend on the target function, a deterministic algorithm to compute such queries will immediately give us the same conclusion starting from SQ algorithms. However, answering a target independent query involves estimating the expectation of a function specified by some Boolean circuit, and no efficient deterministic algorithm is known for this task. We explain why our proof technique cannot accommodate randomized learning algorithms in Section 7.6.

Here, we show that we can indeed prove that SQ algorithms yield lower bound, but our proof here will not be constructive as in the case of CSQ algorithms. Each target independent query asks the oracle an estimate for the expectation of a function, given by some circuit. Thus a $\#\mathsf{P}$ oracle is enough to answer such queries exactly. (Recall that $\#\mathsf{P}$ is the class of counting problems associated with NP-relations.)

The main result of this section follows from Proposition 7.2.8 and a slight modification of the argument used in the proof of Theorem 7.5.7.

Theorem 7.5.9 (SQ Learning Yields Circuit Lower Bounds)**.** *Let $\mathcal{C}$ be a representation class of Boolean functions on $\{-1,1\}^n$. Let ϵ, τ satisfy $\epsilon < 1/2$ and $\tau \leq \min\{\epsilon, 1-2\epsilon\}$. Suppose there exists an algorithm $\mathcal{A}$ that runs in time $T = T(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ that learns $\mathcal{C}^s$ over the uniform distribution to an accuracy of $1-\epsilon$ using at most $Q = Q(n, \frac{1}{\epsilon}, \frac{1}{\tau}, s)$ SQs, each of tolerance τ. Then, at least one of the following conditions hold:*

- *There exists a function (family) $f \in \mathsf{DTIME}(\mathrm{poly}(T, Q, 1/\tau))$ such that for every $c \in \mathcal{C}^s$, we have*
$$\Pr[c(x) \neq f(x)] \geq \tau/4,$$
- *or $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P}$.*

Proof. If $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P}$, we are done. Assume $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{P}$. Thus, we have an efficient deterministic procedure to answer target independent queries. We now follow the proof of Theorem 7.5.7, relying on Proposition 7.2.8. Observe that if we show that the conclusion of Lemma 7.5.6 follows even from an SQ (instead of CSQ) algorithm, we are done, as the rest of the proof of Theorem 7.5.7 does not use $\mathcal{A}$. To show the conclusion of Lemma 7.5.6 starting from

an SQ algorithm, we describe how to obtain a set $\mathbb{G}$ of almost orthogonal functions. As in the original proof, to every correlational query asked by $\mathcal{A}$, f replies with 0. We use the efficient algorithm granted by our assumption to answer target independent queries (each one is specified by some poly(s) size circuit) within τ. Let $g_1, g_2, \cdots, g_k$ be the correlational queries made by algorithm $\mathcal{A}$ before it returns a hypothesis h, and define as before $\mathbb{G} = \{g_i \mid 1 \leq i \leq k\} \cup \{h\}$. The same argument used in the proof of Lemma 7.5.6 applies to $\mathbb{G}$, and the result follows. □

Remark 7. *By using more powerful results it is possible to replace* #P *by smaller complexity classes. In other words, the same argument works for any complexity class that allows us to answer the target independent queries. We omit the details.*

It is known that if P = NP then EXP requires circuits of exponential size.[2] Therefore, we have unconditionally that either $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P}$ or EXP requires large circuits. Comparing this result with our theorem, we observe that under efficient learnability Theorem 7.5.9 implies that either $\mathsf{P}^{\#\mathsf{P}} \not\subseteq \mathsf{P}$ or P contains functions that are average-case hard for $\mathcal{C}$.

7.6 Open problems and further research directions

In this chapter we showed that the existence of deterministic mistake-bounded or exact learning algorithms yield lower bounds as long as the mistake-bound (or queries, respectively) is less than the trivial bound of 2^n. Further, our proofs for these classes work even when the learning algorithms are allowed access to arbitrary oracles. Thus, obtaining a new learning algorithm for a circuit class can be a method to prove new lower bounds against it. An analogous approach was used by Williams [198, 199] to obtain new circuit lower bounds from improved satisfiability algorithms.

Note, however, that our techniques do not yield an explicit lower bound starting from *randomized* learning algorithms. In order to construct a hard function, we must simulate a learning algorithm as a subroutine. If the behavior of the learning algorithm (on a fixed

[2] If P = NP then PH collapses to P. Using a padding argument, it follows that the exponential time hierarchy collapses to EXP, which implies that this class contains functions of exponential circuit complexity by Kannan's theorem.

input) is not deterministic (due to the learning algorithm's internal randomness) then our simulation is not fixed, and so may give different output values starting from the same input. Thus, such learning algorithms will not yield a function. This is also the underlying difficulty in simulating a general SQ algorithm, as the SQ algorithm may ask for an estimate to $\mathrm{E}_x[g(x)]$ for an arbitrary (polynomial-time computable) predicate, which may be hard to approximate deterministically.

We proved that PAC learning algorithms yield a stronger but conditional lower bound (depending on whether PSPACE computations can be sped up by the use of randomness or not). On the other hand, Fortnow and Klivans [70] showed that efficient PAC-learnability of a class $\mathcal{C}$ yields that BPEXP does not have polynomial size circuits from $\mathcal{C}$. Thus, another open question is to extend this line of work and obtain the same conclusion involving $\mathsf{BPSUBEXP}$ instead of BPEXP. As explained above, our "diagonalization" trick to prove lower bounds breaks down in this case, as these algorithms use internal randomness. Interestingly, Volkovich [196] proved that one can obtain a result of this form if a small amount of advice is allowed in the definition of the hard function that is constructed from the learning algorithm.

7.7 Auxiliary results

Here we provide the complete proof of Theorem 7.4.4. We state it again for convenience.

Theorem (Theorem 7.4.4)**.** *Let $\mathcal{C}$ be any concept class and suppose that there exists an algorithm that* PAC *learns $\mathcal{C}$ under the uniform distribution using membership queries when given access to an oracle $\mathcal{O}$ in time $T(n, 1/\epsilon, \log 1/\delta, \mathsf{size}(c))$. Let $L^\star$ be a language that is both downward-self-reducible and $\alpha(n)$-self-correctible. Then, for any constructive function $s : \mathbb{N} \to \mathbb{N}$, at least one of the following conditions hold:*

(i) $L^\star \notin \mathcal{C}^s$*; or*

(ii) $L^\star \in \mathsf{BPTIME}(\mathsf{poly}(T(n, 1/\alpha(n), \log n, s)))^{\mathcal{O}}$.

Proof. If $L^\star \notin \mathcal{C}^s$ then there is nothing to prove. Assume therefore that $L^\star \in \mathcal{C}^s$. In other words, there exists a constant n_0 such that, for every $n > n_0$, there is a concept $c_n \in \mathcal{C}^s_n$ such

that $L^\star_n = c_n$. Let $d(n)$ be a non-decreasing polynomial that upper bounds the number of downward queries necessary to compute $L^\star$ on any input of size n given access to a routine that computes $L^\star$ on inputs of size $n-1$. Since $L^\star$ is self-correctible, there exists an efficient reduction such that, if we can compute $L^\star_n$ correctly on at least a $(1-\alpha(n))$-fraction of the inputs, then we can compute it correctly on every input of size n with probability at least $2/3$. Finally, let $\mathsf{Learner}$ be an algorithm that, when given access to oracle $\mathcal{O}$, is able to learn $\mathcal{C}$ in time $T(n, 1/\epsilon, \log 1/\delta, \mathsf{size}(c))$, where $\mathsf{size}(c)$ is an upper bound on the size of the unknown concept.

We need to prove that $L^\star \in \mathsf{BPTIME}(\mathsf{poly}(T(n, 1/\alpha(n), \log n, s)))^{\mathcal{O}}$. Given an input x of size n, we use $\mathsf{Learner}$ and the special properties of language $L^\star$ to "learn" how to compute $L^\star$ on every instance of size at most n. More specifically, for any integer k, given a procedure A_k that decides $L^\star$ on any instance of size k (with high probability), we use $\mathsf{Learner}$ together with the downward-self-reducibility of $L^\star$ and the fact that this language is self-correctible to obtain a procedure A_{k+1} that decides $L^\star$ on any instance of size $k+1$ (with high probability).

The execution of the algorithm for $L^\star$ proceed as follows. First, it starts with a procedure A_{n_0} that can be implemented by a lookup-table algorithm (recall that n_0 is a constant). Now we explain in more details how to go from, say, A_{n_0} to A_{n_0+1}. We simulate $\mathsf{Learner}$ pretending that the unknown concept is $c_{n_0+1} = L^\star_{n_0+1}$. If at some point the learning algorithm queries the value of $c_{n_0+1}(w)$ on some input w of size n_0+1, we use A_{n_0} together with the downward-self-reducibility property of $L^\star$ to provide an appropriate answer. If $\mathsf{Learner}$ queries its oracle $\mathcal{O}$, we provide the answer using our own oracle $\mathcal{O}$. After finishing its computation, the learning algorithm outputs a deterministic hypothesis h_{n_0+1} that is ϵ-close to c_{n_0+1} with high probability. We use the fact that $L^\star$ is self-correctible to obtain from h_{n_0+1} a procedure $\tilde{A}_{n_0+1}$ that is correct on every input of size n_0+1 with probability at least $2/3$. Finally, using standard amplification techniques, it is possible to get from $\tilde{A}_{n_0+1}$ a procedure A_{n_0+1} that is correct on every input with high probability. By repeating this process at most n stages, we obtain a procedure A_n and output $A_n(x)$. Let $\mathcal{A}$ be the algorithm that runs as described. The formal description of $\mathcal{A}$ is presented in Algorithm 3.

Algorithm 3 Description of Algorithm $\mathcal{A}$ that computes a hard function using PAC learner

Require:

Input: A string x of size n (and oracle access to $\mathcal{O}$).

Output: The value $L^\star(x)$ (with high probability).

1: Start with a "lookup-table" routine A_{n_0} that computes L correctly on all inputs of size n_0.

2: **for** $k = n_0 + 1$ to n **do**

3: Run Learner with parameters k, $\epsilon = \alpha(n)$, $\delta = 1/20n$, and $\mathsf{size}(c) = s(k)$ (here we use the fact that $s(\cdot)$ is constructible). Whenever Learner asks for the value $c_k(w)$ of some example w of size k, use routine A_{k-1} and the downward self-reducibility of $L^\star$ to compute a guess for $c_k(x) = L(x)$. Since $\mathcal{A}$ has oracle access to $\mathcal{O}$, any query to this oracle made by Learner can also be answered efficiently. When the learning algorithm finishes its computation, it outputs with probability at least $1 - \delta$ a deterministic hypothesis h_k that is ϵ-close to c_k (note that h_k does not have access to $\mathcal{O}$).

4: Since $\epsilon = \alpha(n)$ and $L^\star$ is $\alpha(n)$-self-correctible, $\mathcal{A}$ uses h_k and the self-correctibility of $L^\star$ to get a routine $\tilde{A}_k$ such that for any input w of size k, $\tilde{A}_k(w) = L^\star(w)$ with probability at least $2/3$. By running $\tilde{A}_k$ at most $O(\log 1/\gamma)$ times and taking a majority vote, it follows from standard Chernoff bounds that we obtain a routine A_k that is incorrect on any input with probability at most γ. We set $\gamma = 1/(20t_n d(n)n)$, where $t_n = T(n, 1/\alpha(n), \log 20n, s)$.

5: **end for**

6: **return** $A_n(x)$.

First we argue that $\mathcal{A}$ computes $L^\star(x)$ correctly with high probability, then we upper bound its running time.

Claim 7.7.1. *For any input x, $\mathcal{A}$ outputs $L^\star(x)$ with probability at least $2/3$.*

Proof. Note that, for each stage k, $\mathcal{A}$ fails to obtain a good routine A_k only if:

- At least one the at most $t_n \cdot d(n)$ downward queries answered by A_{k-1} is incorrect. It follows by a union bound that this happens with probability at most $t_n \cdot d(n) \cdot \gamma = 1/20n$.
- Algorithm Learner does not output a good hypothesis. This also happens with probability at most $\delta = 1/20n$.

Overall, for each stage k, we fail to obtain a good algorithm A_k with probability no more than $1/10n$. Since there are at most n stages, A_n fails to compute $L^\star(x)$ with probability at most $1/10 + \gamma \leq 1/3$. □

Claim 7.7.2. *Given oracle access to $\mathcal{O}$, algorithm A runs in randomized time at most* $\mathrm{poly}(T(n, 1/\alpha(n), \log n, s)))$.

Proof. First we upper bound the running time of each procedure A_k. Observe that A_k uses $\tilde{A}_k$, which is obtained from h_k. Recall that the running time of h_k is bounded by the running time of Learner, which is at most $t(k, 1/\alpha(n), \log 20n, s(k)) \leq T(n, 1/\alpha(n), \log 20n, s) = t_n$, since both $s(\cdot)$ and $t(\cdot)$ are non-decreasing[3]. Further, to obtain $\tilde{A}_k$ we use the self-correctibility of $L^\star$, which is implemented by a polynomial-time reduction. In other words, $\tilde{A}_k$ runs in time $O(t_n^a)$ for some constant a. Finally, the amplification step that is used when we go from $\tilde{A}_k$ to A_k only needs to run $\tilde{A}_k$ for $O(\log 1/\gamma) = O(\log(20t_n \cdot d(n) \cdot n))$ times, which implies that the overall time complexity of A_k, for any $1 \leq k \leq n$, is upper bounded by $O(t_n^a \cdot \log(20t_n \cdot d(n) \cdot n)) = O(t_n^b)$ for some constant b (recall that $d(\cdot)$ is a polynomial).

Now we upper bound the overall running time of algorithm $\mathcal{A}$. Each stage k consists of simulating algorithm Learner for at most $t(k, 1/\alpha(n), \log 20n, s(k)) \leq t_n$ steps. In the worst-case, each step may involve a membership query to the unknown concept, which translates to at most $d(n)$ downward queries to A_{k-1}. Since A_{k-1} runs in time $O(t_n^b)$, the overall time complexity of each stage is at most $O(t_n \cdot d(n) \cdot t_n^b) = O(t_n^c)$ for some constant c. There are no more than n stages. It follows that, given oracle access to $\mathcal{O}$, algorithm $\mathcal{A}$ runs in randomized time $\mathrm{poly}(T(n, 1/\alpha(n), \log n, s)))$. □

□

[3]We have implicitly used the standard fact that any PAC learning algorithm can be efficiently converted into an algorithm that has a logarithmic dependence on $1/\delta$. A proof of this result can be found on Kearns and Vazirani [117] textbook.

Chapter 8

Satisfiability algorithms, useful properties, and lower bounds

8.1 Background, results, and organization

This chapter is concerned with two research directions in theoretical computer science: the design of nontrivial algorithms for difficult computational tasks, and the search for unconditional proofs that some natural computational problems are inherently hard (more specifically, do not admit polynomial size circuits).

Perhaps surprisingly, these problems are deeply related. For instance, it follows from the work of Karp and Lipton [115] (attributed to Meyer) that if 3-SAT admits a polynomial time algorithm, then there are problems solved in exponential time that cannot be computed by polynomial size circuits. On the other hand, it is known that constructive proofs of circuit lower bounds lead to algorithms breaking exponentially hard pseudorandom generators that are conjectured to exist (Razborov and Rudich [154]).

The last decade has produced several additional *transference theorems*[1] of this form, under many different algorithmic frameworks. For instance, the existence of subexponential time learning algorithms for a class of functions $\mathcal{C}$ leads to circuit lower bounds against $\mathcal{C}$

[1] In other words, these theorems show that one can transform an algorithmic result into a circuit lower bound, i.e., they allow us to transfer a result from one area to another.

(Fortnow and Klivans [70]). In a different domain, it is known that the design of subexponential time deterministic algorithms for problems with efficient randomized algorithms implies circuit lower bounds that have eluded researchers for decades (Kabanets and Impagliazzo [110]). More recently, it has been shown that new circuit lower bounds can be obtained from efficient compression algorithms (Chen et al. [49]), not to mention the connection between satisfiability algorithms and circuit lower bounds (Williams [198], [199], [201]). Several additional results have appeared in the literature ([120], [1], [3], [21], [92], [122], among others). For a gentle introduction to some of these connections, see Santhanam [168].

It turns out that the connection between algorithms and circuit lower bounds ("transference theorems") can be used to prove *new* circuit lower bounds that had resisted the use of more direct approaches for decades. Let $\mathcal{C}$ be a class of circuits, such as $\mathsf{AC}^0, \mathsf{TC}^0, \mathsf{NC}^1$, etc. We say that a satisfiability algorithm for $\mathcal{C}$ is *nontrivial* if it runs in time time $2^n/s(n)$, for some function $s(n) \gg \text{poly}(n)$. Building on work done by many researchers, Williams ([199], [198]) proved the following transference theorem: the existence of a nontrivial $\mathcal{C}$-SAT algorithm implies $\mathsf{NEXP} \not\subseteq \mathcal{C}[\text{poly}]$. In other words, faster satisfiability algorithms lead to languages computed in nondeterminstic exponential time that cannot be computed by polynomial size circuits from $\mathcal{C}$.

Most importantly, by designing a new ACC-SAT algorithm, Williams [199] was able to obtain a circuit lower bound for the circuit class ACC.[2] Moreover, this is the *only* known proof of this result. Other approaches that have been proposed are also based on the design of new ACC algorithms (Chen et al. [49]).

Can we extend this technique to prove stronger circuit lower bounds? Is there any connection between Williams' transference theorem and other similar results discussed before? This chapter is motivated by these questions. We break this introductory section into two parts. The first part is a fast-paced introduction to some results connecting algorithms to circuit lower bounds. After this (non-exhaustive) introduction to the literature, we discuss

[2]This is the class of languages computed by polynomial size constant-depth circuits consisting of AND, OR, NOT and MOD_m gates (for a fixed integer $m \in \mathbb{N}$). Every gate other than NOT is allowed to have unbounded fan-in.

our contributions, which extend and simplify a few of these connections.

We stress that our focus here is on *generic* connections between faster algorithms and circuit lower bounds, instead of particular techniques that have found applications in both areas (Fourier representation of Boolean functions [129], satisfiability coding lemma [151], random restriction method [49], etc.).

8.1.1 A brief introduction to transference theorems

Satisfiability algorithms and circuit lower bounds. The connection between algorithms for hard problems and circuit lower bounds has been known for decades. More precisely, a collapse theorem attributed to Meyer [115] states that if $\mathsf{EXP} \subseteq \mathsf{P/poly}$ then $\mathsf{EXP} = \Sigma_2^p$ (recall this is the second level of PH, the polynomial time hierarchy). On the other hand, it is not hard to prove that if $\mathsf{P} = \mathsf{NP}$ then $\mathsf{P} = \Sigma_2^p = \mathsf{PH}$. Together, the assumptions that there are efficient algorithms for NP-complete problems and that every problem in EXP admits polynomial size circuits lead to $\mathsf{P} = \mathsf{EXP}$, a contradiction to the deterministic time hierarchy theorem. In other words, if there exists efficient algorithms for 3-SAT, it must be the case that $\mathsf{EXP} \not\subseteq \mathsf{P/poly}$.[3] Similar transference results can be obtained from the assumption that there are subexponential time algorithms for 3-SAT (i.e., with running time $2^{n^{o(1)}}$).

The existence of such algorithms is a very strong assumption. The best known algorithms for k-SAT run in time $2^{n(1-\delta(k))}$, for some fixed constant $\delta(k) > 0$ that goes to zero as k goes to infinity (cf. Dantsin and Hirsch [55]). These algorithms offer an exponential improvement over the trivial running time $\tilde{O}(2^n)$. If we only require the running time to be faster than exhaustive search ("nontrivial"), then improved algorithms are known for many interesting circuit classes (see for instance [167], [49], [173], [107], [27], [106]). For an introduction to some of these algorithms, see Schneider [170].

It makes sense therefore to investigate more refined versions of the transference theorem for satisfiability algorithms. A result in this directions was obtained by Williams [198]: he showed that the existence of nontrivial algorithms deciding the satisfiability of polynomial

[3] Using the fact that $\mathsf{P} = \mathsf{PH}$ implies the collapse of the exponential time hierarchy to EXP, an even stronger consequence can be obtained. We omit the details.

size circuits is enough to imply $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. Unfortunately, $\mathsf{P/poly}$ is a very broad class, and the algorithms mentioned before do not work or have trivial running time on such circuits.

In a follow-up work, Williams [199] extended his techniques from [198] to prove a more general result that holds for other circuit classes as well.

Proposition 8.1.1 ("SAT algorithms yield circuit lower bounds, I" [199])**.**
Let $\mathcal{C}$ be a class of circuit families that is closed under composition (the composition of two circuits from $\mathcal{C}$ is also in $\mathcal{C}$) and contains AC^0. There is a $k > 0$ such that, if satisfiability of $\mathcal{C}$-circuits with n variables and n^c size can be solved in $O(2^n/n^k)$ time for every c, then $\mathsf{NEXP} \not\subseteq \mathcal{C}[\mathrm{poly}(n)]$.

In addition, he provided a nontrivial algorithm for $\mathsf{ACC}[2^{n^\delta}]$ (the class of ACC circuits of size 2^{n^δ}), where $\delta = \delta(d, m) > 0$ depends on the depth of the circuit and the modulo gate. Altogether, these results imply the following circuit lower bound.

Corollary 8.1.2. $\mathsf{NEXP} \not\subseteq \mathsf{ACC}$.

Subsequent work of Williams [201] extended these techniques to prove the following stronger transference theorem, which provides better circuit lower bounds.[4]

Proposition 8.1.3 ("SAT algorithms yield circuit lower bounds, II" [201])**.**
Let $\mathcal{C}$ be a class of circuit families that is closed under composition and contains AC^0. There is a $k > 0$ such that, if satisfiability of $\mathcal{C}$-circuits with n variables and $n^{\log^c n}$ size can be solved in $O(2^n/n^k)$ time for every c, then $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathcal{C}[n^{\log n}]$.

Moreover, the following strengthening of Corollary 8.1.2 is proven in the same paper (the first statement is implicit in his proof).

Corollary 8.1.4. $\mathsf{E} \not\subseteq \mathsf{ACC}[n^{\log n}]$ *or* $\mathsf{Quasi\text{-}NP} \cap \mathsf{i.o.Quasi\text{-}coNP} \not\subseteq \mathsf{ACC}[n^{\log n}]$. *In particular,* $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathsf{ACC}[n^{\log n}]$.

[4] We use $\mathsf{NE} \cap \mathsf{i.o.coNE}$ instead of $\mathsf{NE} \cap \mathsf{coNE}$ in the statement of Proposition 8.1.3 because the proof described in [201] requires this extra condition [200].

The proof of these transference theorems has been simplified since then. For instance, Santhanam and Williams [166] employed self-reduction (cf. Allender and Koucký [10]) to obtain an equivalent circuit in a smaller circuit class given an arbitrary NC^1 circuit (under appropriate assumptions). This simplifies one of the main technical lemmas from [199]. (We introduce another technique which yields a stronger transference theorem for satisfiability in Section 8.1.2.)

Constructivity and circuit lower bounds. There are at least three important barriers to circuit lower bound proofs: relativization (Baker, Gill, and Solovay [24]), natural proofs (Razborov and Rudich [154]), and algebrization (Aaronson and Wigderson [2]). Roughly speaking, these barriers can be interpreted as follows: some proof methods are too general, and if a lower bound can be obtained by one of such techniques alone, then we get a contradiction to some known result or a widely believed conjecture.[5] As explained by Williams [199], his lower bound proof combines several methods used in modern complexity theory, and each one avoids a particular barrier.[6]

It was proven by Razborov and Rudich that most of the circuit lower bound proofs known at the time proceeded (at least implicitly) as follows. There is a circuit class $\mathcal{C}$ (say, AC^0) that one wants to separate from a complexity class Γ (say, P). In order to do that, one defines a property $\mathcal{P}$ of Boolean functions (i.e., a subset of all Boolean functions), and prove that *no* function in $\mathcal{C}$ satisfies $\mathcal{P}$, while there exists some hard function $h \in \Gamma$ for which $\mathcal{P}(h) = 1$ (in this case, we say that $\mathcal{P}$ is *useful* against $\mathcal{C}$). For instance, every AC^0 function simplifies after an appropriate random restriction ([75], [205], [98]), while the parity function is still as hard as before.

As it turns out, for the property $\mathcal{P}$ defined in these proofs, there is an efficient algorithm $\mathcal{A}$ (with respect to the size of the truth-table of f) that is able to decide whether $\mathcal{P}(f) = 1$. Such properties are referred to as *constructive* properties. In addition, it is usually the case

[5] These barriers can also be interpreted as independence results for some formal theories ([20], [158], [105]).

[6] We stress, however, that there is no widely believed conjecture that leads to pseudorandom function families in ACC, and this is an interesting open problem. As far as we know, there may exist a natural proof that $\mathsf{P} \not\subseteq \mathsf{ACC}$.

that a random function satisfies $\mathcal{P}$ with non-negligible probability ($\mathcal{P}$ satisfies the *denseness* condition). These two conditions imply that $\mathcal{A}$ can be used to distinguish a function in $\mathcal{C}$ from a random function. Put another way, if there exists a proof of this form that $\Gamma \not\subseteq \mathcal{C}$, then there is no pseudorandom function family in $\mathcal{C}$.

However, if some number-theoretic problems are exponentially hard on average (an assumption believed to be true by many researchers), then there are pseudorandom functions in circuit classes as small as TC^0_4 (Naor and Reingold [142], Krause and Lucks [127]). As a consequence, such proofs (dubbed *natural proofs* in [154]) are not expected to prove separations for more expressive circuit classes. Unfortunately, most (if not all) known combinatorial proofs of circuit lower bounds implicitly define such properties, and this explains the lack of significant progress obtained so far for more general classes of circuits using these techniques only. The interested reader is referred to Chow [50] and Rudich [164] for further developments.

As a consequence, any circuit lower bound proof for more expressive classes must violate either the denseness or the constructivity condition. Williams [201] shed light into this problem, by proving that any separation of the form $\mathsf{NEXP} \not\subseteq \mathcal{C}$ is actually equivalent to exhibiting a *constructive* property $\mathcal{P}$ that is useful against $\mathcal{C}$.

Proposition 8.1.5 ("Constructivity is unavoidable", **informal** [201])**.**
Let $\mathcal{C}$ be a typical circuit class. Then $\mathsf{NEXP} \not\subseteq \mathcal{C}$ if and only if there exists a constructive property $\mathcal{P}$ that is useful against $\mathcal{C}$.

In other words, any lower bound proof against NEXP implies the existence of a property that is both useful and constructive, but not necessarily dense. (As we explain later in the text, $\mathcal{P}$ is actually computed with a small amount of advice. We clarify this point in Section 8.1.2, where we discuss some extensions of the connection between useful properties and circuit lower bounds.)

Additional transference theorems. As alluded to earlier, several additional transference theorems of the form "faster algorithms yield circuit lower bounds" have been discovered. In the next few paragraphs we describe some of these results in more detail. We focus on learning algorithms, derandomization, and algorithms for string compression.

Derandomization. There is a strong connection between the existence of pseudorandom generators and circuit lower bounds (see [111], [188]). Furthermore, in some contexts it is possible to show that pseudorandom generators are necessary in order to derandomize probabilistic algorithms (Goldreich [85]).

For the randomized complexity class MA, it is known that any derandomization (such as MA $\subseteq$ NSUBEXP) implies superpolynomial circuit lower bounds for NEXP (Impagliazzo, Kabanets and Wigderson [104]). Subsequent work of Kabanets and Impagliazzo [110] shows that even the derandomization of a single, specific problem in BPP leads to some circuit lower bounds. More precisely, let PIT be the language consisting of all arithmetic circuits that compute the zero polynomial over $\mathbb{Z}$, and PERM be the problem of computing the permanent of integer matrices. We use SIZE[poly] to denote the set of languages computed by polynomial size Boolean circuits. Similarly, let ASIZE[poly] be the family of languages computed by arithmetic circuits of polynomial size over $\mathbb{Z}$.

Proposition 8.1.6 ("Derandomization yields circuit lower bounds" [110])**.**
If PIT $\in$ NSUBEXP*, then at least one of the following results hold:*

(i) NEXP $\not\subseteq$ SIZE[poly]*; or*

(ii) PERM $\not\subseteq$ ASIZE[poly].

Aaronson and van Melkebeek [1] proved a parameterized version of the result, in addition to showing that NEXP $\cap$ coNEXP can be used in place of NEXP. Another extension appears in Kinne, van Melkebeek and Shaltiel [120].

Learning algorithms. Fortnow and Klivans [70] were the first to investigate more systematically the connections between learning algorithms and circuit lower bounds, following results obtained by Impagliazzo and Wigderson [102]. Recall that a learning algorithm $\mathcal{A}$ is given restricted access to a fixed but arbitrary function f from a class of functions $\mathcal{C}$, and it should output a hypothesis h that is as close to f as possible. Distinct learning models provide difference access mechanisms to f, and impose specific requirements over h (h should be close to f, $h \equiv f$, etc.) and $\mathcal{A}$ (learner is randomized, deterministic, etc).

An exact learning algorithm is a *deterministic* algorithm that has access to a membership query oracle MQ^f and an equivalence query oracle EQ^f, and it is required to output a hypothesis h which agrees with f over all inputs.[7]

Proposition 8.1.7 ("Learning yields circuit lower bounds" [70])**.**
Let $\mathcal{C}$ be a circuit class. If there exists a subexponential time exact learning algorithm for $\mathcal{C}$, then $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathcal{C}$.

The original proof of Proposition 8.1.7 relies on many complexity theoretic results. Subsequent work done by Harkins and Hitchcock [92] strengthened the conclusion to $\mathsf{EXP} \not\subseteq \mathcal{C}$. Finally, Klivans, Kothari and Oliveira [122] used a very simple argument to prove the essentially optimal result that exact learning algorithms for $\mathcal{C}[s(n)]$ running in time $t(n)$ lead to a circuit lower bound of the form $\mathsf{DTIME}[\mathrm{poly}(t(n))] \not\subseteq \mathcal{C}[s(n)]$.[8]

Weaker results have been obtained for *randomized* learning algorithms (a formal definition of the model is discussed in Section 8.5.3). Efficient PAC learning algorithms are known to lead to circuit lower bounds against BPEXP, the exponential time analogue of BPP [70]. A slightly stronger result was obtained by Klivans et al. [122], but the underlying techniques do not provide interesting results for randomized subexponential time algorithms. Volkovich [196] proved that if a small amount of advice is allowed in the definition of the hard function, then one can obtain strong lower bounds from randomized learning algorithms (see [196] for further details). Obtaining better lower bounds from randomized learning algorithms without advice remains an interesting open problem.

Truth-table compression. More recently, Chen et al. [49] considered the problem of designing efficient algorithms that obtain nontrivial compression of strings representing truth-tables from a circuit class $\mathcal{C}$. In other words, given a string $tt(f_n) \in \{0,1\}^N$, where $f_n : \{0,1\}^n \to \{0,1\}$ is a function from $\mathcal{C} \subseteq \mathsf{P/poly}$ and $N = 2^n$, a *compression algorithm* is required to run in time $\mathrm{poly}(N)$ and output a circuit C over n inputs and size $\ll 2^n/n$ such that $tt(C) = tt(f)$. In the same paper, they observed that several circuit lower bound proofs

[7] On input $x \in \{0,1\}^n$, $\mathsf{MQ}^f(x)$ returns $f(x)$. On input a circuit c, EQ^f outputs "yes" if $c \equiv f$, otherwise it outputs an arbitrary input z such that $c(z) \neq f(z)$.

[8] The same result was obtained independently by Russell Impagliazzo and Valentine Kabanets [112].

that rely on the method of random restrictions yield efficient compression algorithms. On the other hand, they obtained the following transference theorem.

Proposition 8.1.8 ("Compression leads to circuit lower bounds" [49])**.**
Let $\mathcal{C}$ be a circuit class. Suppose that for every $c \in \mathbb{N}$ there is a deterministic polynomial-time algorithm that compresses a given truth table of an n-variate Boolean function $f \in \mathcal{C}[n^c]$ to an equivalent circuit of size $o(2^n/n)$. Then $\mathsf{NEXP} \not\subseteq \mathcal{C}$.

It follows from Proposition 8.1.8 that designing a compression algorithm for ACC would provide an alternative proof of Corollary 8.1.2. This is left as an interesting open problem by [49].

8.1.2 Our results

Lower bounds from satisfiability algorithms for low depth circuits. Let TC^0_2 denote the class of languages admitting polynomial size circuits of depth two with gates that compute arbitrary linear threshold functions. Perhaps surprisingly, it is consistent with our current knowledge that $\mathsf{NEXP} \subseteq \mathsf{TC}^0_2$. It makes sense therefore to investigate whether the techniques used in the proof of Corollary 8.1.2 can be helpful in proving separations against bounded-depth circuit classes of this form.

A more refined version of Proposition 8.1.1 discussed in [199] shows that circuit lower bounds for circuits of depth d follow from satisfiability algorithms for depth $2d + O(1)$. We prove that it is possible to obtain a tight transference theorem for satisfiability algorithms for constant-depth circuits. Let $\mathcal{C}_d$ be a circuit class consisting of circuits of depth d, and g be an arbitrary function. We write $g[k] \circ \mathcal{C}_d$ to denote the class of functions computed by circuits of depth $d + 1$ consisting of a top layer gate g of fan-in k that is fed by k circuits from $\mathcal{C}_d$.

Theorem 8.1.9 ("SAT algorithms for depth $d+2$ yield circuit lower bounds for depth d")**.**
Let $\mathcal{C}$ be a reasonable circuit class. If there exists a nontrivial satisfiability algorithm for $\mathsf{AND}[3] \circ \mathsf{OR}[2] \circ \mathcal{C}_d[\mathrm{poly}]$, then $\mathsf{NEXP} \not\subseteq \mathcal{C}_d[\mathrm{poly}]$.

We define reasonable circuit classes in Section 8.2. This result can be obtained through an extension of the original technique used by Williams [198]. In particular, our presenta-

tion avoids the technical details from [199]. The proof of Theorem 8.1.9 and some additional remarks are presented in Section 8.3. (A similar result is described in Jahanjou, Miles and Viola [108], but their approach is different.)

Useful properties and circuit lower bounds. We investigate more closely the relation between circuit lower bounds and useful properties (Proposition 8.1.5). For a uniform complexity class Γ (such as P, NP, etc), we say that a property of Boolean functions $\mathcal{P}$ is a Γ-property if it can be decided in Γ. We use $\Gamma/s(m)$ to denote the corresponding complexity class with advice of size $s(m)$, where m is the size of the input. Recall that a property is useful against $\mathcal{C}$ if it distinguishes some hard function from all functions in $\mathcal{C}$ (a formal definition is presented in Section 8.2).

First, we notice that nondeterminism is of no use in the context of useful properties, which is a somewhat surprising result. This result has been independently observed by other authors (cf. Allender [11]), and we include a self-contained proof here using our notation for convenience of the reader.[9]

Theorem 8.1.10 ("NP-property yields P-property")**.**
Let $\mathcal{C}$ be a circuit class. If there exists a NP-property that is useful against $\mathcal{C}[\mathrm{poly}]$, then there is a P-property that is useful against $\mathcal{C}[\mathrm{poly}]$.

We discuss now in more detail the connection discovered by Williams (Proposition 8.1.5) between constructive useful properties (P-properties under our notation) and circuit lower bounds. It turns out that the statement of Proposition 8.1.5 requires a broader definition, one for which the algorithm deciding the property is allowed inputs of arbitrary size instead of size $N = 2^n$, where $n \in \mathbb{N}$. Put another way, the algorithm receives any string as input, and is allowed to parse its input size as $2^n + k$. Now it is free to interpret k as an advice string of length $\log N$. We clarify this issue here, and observe that Theorem 8.1.10 together with standard techniques imply the following characterization of NEXP circuit lower bounds.[10]

[9] The statement presented next follows from a slightly more general result (Proposition 8.4.1) proved in Section 8.4.

[10] We stress that in this chapter any algorithm that decides a property of Boolean functions works over

Theorem 8.1.11 ("Equivalence between NEXP lower bounds and useful properties")**.**
Let $\mathcal{C}$ be a circuit class. Then $\mathsf{NEXP} \not\subseteq \mathcal{C}[\mathrm{poly}]$ *if and only if there exists a* $\mathsf{P}/\log N$*-property that is useful against* $\mathcal{C}[\mathrm{poly}]$*.*

It makes sense therefore to investigate whether there exists an equivalence between useful properties computed without advice and circuit lower bounds. The following result holds.

Theorem 8.1.12 ("$\mathsf{NE} \cap \mathsf{coNE}$ lower bounds and useful properties")**.**
Let $\mathcal{C}$ be a circuit class. The following holds:

(*i*) *If* $\mathsf{NE} \cap \mathsf{coNE} \not\subseteq \mathcal{C}[\mathrm{poly}]$ *then there is a* P*-property that is useful against* $\mathcal{C}[\mathrm{poly}]$*.*

(*ii*) *If for every* $c \in \mathbb{N}$ *there exists a* P*-property that is useful against* $\mathcal{C}[n^{\log^c n}]$*, then* $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathcal{C}[n^{\log n}]$*.*

One direction follows from Theorem 8.1.10, while the other is implicit in the work of Williams [201]. Given these results, the following conjecture seems plausible.

Conjecture 8.1.13 ("$\mathsf{NE} \cap \mathsf{coNE}$ lower bounds versus useful properties?")**.**
Let $\mathcal{C}$ be a circuit class. Then $\mathsf{NE} \cap \mathsf{coNE} \not\subseteq \mathcal{C}[\mathrm{poly}]$ *if and only if there exists a* P*-property that is useful against* $\mathcal{C}[\mathrm{poly}]$*.*

We discuss how this conjecture relates to Williams' program for circuit lower bounds in Section 8.4.1. The results related to useful properties appear in Section 8.4.

Applications and further connetions. It is possible to use the results mentioned above to prove a few propositions stated in Section 8.1.1. Further, we observe that several transference theorems are in fact *connected*, and improvements in one framework propagates to other results.

The first application that we discuss is for compression algorithms, as investigated by Chen et al. [49]. Observe that Proposition 8.1.8 shows circuit lower bounds for NEXP from exact compression of truth-tables of polynomial size circuits. As mentioned in the same paper, their result can be extended to show that even *lossy* compression algorithms

strings of length $N = 2^n$, where $n \in \mathbb{N}$.

lead to circuit lower bounds. We flesh out the details here, and obtain a slightly stronger connection as well.

We say that a circuit class $\mathcal{C}$ admits *lossy compression* algorithms if there exists an efficient algorithm $\mathcal{A}$ (over inputs of size $N = 2^n$) that when given as input a truth-table $tt(f)$ from $\mathcal{C}$, where $f : \{0,1\}^n \to \{0,1\}$, outputs a circuit C of size $o(2^n/n)$ such that $\Pr_x[C(x) = f(x)] \geq .51$. A more general definition is discussed in Section 8.5.1.

Theorem 8.1.14 ("Circuit lower bounds from lossy compression")**.**
Let $\mathcal{C}$ be a circuit class. The following results hold.

(*i*) *If for every $c \in \mathbb{N}$ there exists a lossy compression algorithm for $\mathcal{C}[n^c]$, then* $\mathsf{NEXP} \not\subseteq \mathcal{C}[\mathrm{poly}(n)]$.

(*ii*) *If for every $c \in \mathbb{N}$ there exists a lossy compression algorithm for $\mathcal{C}[n^{\log^c n}]$, then* $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathcal{C}[n^{\log n}]$.

In particular, any efficient algorithm for lossy compression of strings is either trivial on infinitely many input strings represented by truth-tables from TC^0_2 (i.e., does not provide a lossy encoding of significantly smaller size), or a certain circuit lower bound holds. Theorem 8.1.14 follows from an easy application of Theorem 8.1.11, and its proof is presented in Section 8.5.1.

Next we observe that Proposition 8.1.6 ("derandomization yield circuit lower bounds") follows from the transference theorem for satisfiability. More precisely, Theorem 8.1.9 extends to slightly more general algorithms, an observation that we discuss in more detail in Section 8.3.1. Using this generalization, it is possible to prove that if Proposition 8.1.6 is false, then a contradiction can be obtained. This proof is presented in Section 8.5.2.

In the context of learning algorithms, some extensions of the main result of Fortnow and Klivans [70] for exact learning (Proposition 8.1.7) follow easily from results for useful properties (Theorems 8.1.11 and 8.1.12). In addition, it is not hard to show that even subexponential time randomized learning leads to useful properties decided by efficient randomized algorithms.

Theorem 8.1.15 ("Useful properties and learning algorithms")**.**
Let $\mathcal{C} = \mathcal{C}[\mathrm{poly}]$ be a circuit class. If there exists a subexponential time randomized PAC

learning algorithm for $\mathcal{C}$, then there exists a (promise)coRP*-property that is useful against $\mathcal{C}$.*

These transference theorems can be obtained by interpreting learning algorithms as lossy compression schemes, or by relying directly on Theorems 8.1.11 and 8.1.12. These results are discussed in more detail in Section 8.5.3.

Overall, these observations show that an improvement of a transference theorem in one framework leads to similar improvements in other frameworks. For instance, a proof of Conjecture 8.1.13 implies many interesting results of the form "nontrivial algorithms yield circuit lower bounds". More precisely, it immediately implies new transference theorems for both (lossy) compression and satisfiability algorithms, and an alternative proof of the extension of Proposition 8.1.6 obtained by Aaronson and van Melkebeek [1]. Moreover, a direct improvement of the transference theorems for satisfiability is likely to imply a similar strengthening of Proposition 8.1.6.

8.1.2.1 An overview of the results

For convenience of the reader, Figure 8.1 summarizes the relations between algorithms and circuit lower bounds discussed in this chapter.

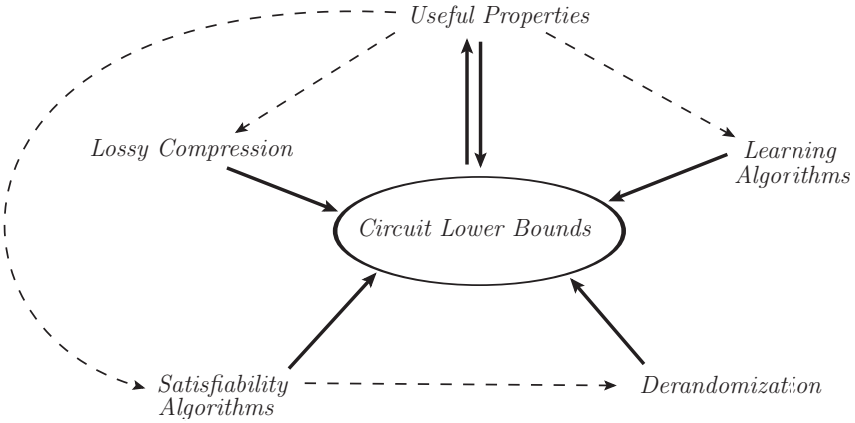

Figure 8.1: Bold arrows represent transference theorems, while a dotted arrow from A to B indicates that an improvement of the transference theorem for A implies a similar improvement of the transference theorem for B.

8.2 Preliminaries and notation

We assume familiarity with basic notions from computational complexity theory. The reader is referred to Arora and Barak [19] and Goldreich [84] for more details. For convenience, we postpone some definitions that are specific to a particular section of the chapter to that corresponding section.

We say that h is a family of Boolean functions if $h = \{h_n\}_{n\in\mathbb{N}}$, where each $h_n \colon \{0,1\}^n \to \{0,1\}$. Recall that any such family corresponds to a language $L \subseteq \{0,1\}^*$, and that L_n denotes $L \cap \{0,1\}^n$.

We will use Γ to denote uniform complexity classes such as P, coRP and NP. Sometimes we will extend these complexity classes to the corresponding classes with advice of size $a(m)$, where m is the input size. In this case, we use $\Gamma/a(m)$. A language L is in $\mathsf{i.o.}\Gamma$ if there exists $L' \in \Gamma$ such that $L_n = L'_n$ for infinitely many values of n.

Following [201], we say that a circuit class $\mathcal{C}$ is *typical* if $\mathcal{C} \in \{\mathsf{AC}^0, \mathsf{ACC}, \mathsf{TC}^0, \mathsf{NC}^1, \mathsf{P/poly}\}$. The results stated for typical classes hold for more general circuit classes. We use $\mathsf{SIZE}[s(n)]$ to denote the family of functions computed by circuits of size $s(n)$. Similarly, $\mathsf{ASIZE}[s(n)]$ denotes the family of functions computed by arithmetic circuits of size $s(n)$. Although each circuit class corresponds to a set of languages, we may abuse notation and say that a given circuit D is from $\mathcal{C}$. In general, for any circuit class $\mathcal{C}$, let $\mathcal{C}_d[s(n)]$ be the family of functions computed by circuits from $\mathcal{C}$ of depth d and size $s(n)$, where the size of a circuit is the number of gates in the circuit. If for convenience we omit $s(n)$, assume the circuits are of polynomial size. For instance, $\mathsf{TC}^0_2[n^2]$ corresponds to the class of languages computed by circuits of depth-two with $O(n^2)$ gates, each one corresponding to some linear threshold function. All circuit classes considered here are *non-uniform*. If we mention a circuit D of size $s(n)$ without attributing it to a specific circuit class, assume it is composed of AND, OR and NOT gates of fan-in at most two.

In order to prove a tight transference theorem for some circuit classes, we make the following definition.

Definition 8.2.1. *A circuit class $\mathcal{C}$ is* reasonable *if:*

(*i*) *The constant zero function $f : \{0,1\}^n \to \{0,1\}$ with $f(x) = 0$ for every input x is in*

$\mathcal{C}$.

(*ii*) *For every function $g \in \mathcal{C}$, the function $\bar{g} = \mathsf{NOT}(g)$ is in $\mathcal{C}$, i.e., $\mathcal{C}$ is closed under complementation. In addition, there is an efficient algorithm that, given the description of a circuit computing g, outputs a circuit from $\mathcal{C}$ of the same size computing $\bar{g}$.*

(*iii*) *The gates of circuits from $\mathcal{C}$ may have direct access to constant inputs 0 and 1 in addition to the input variables and their negations.*[11]

(*iv*) *Any language in $\mathcal{C}[\mathrm{poly}(n)]$ is in $\mathsf{P/poly}$.*

The results that are stated for reasonable classes hold for more general circuit classes, but for simplicity we stick with this definition. In any case, most circuit classes are reasonable (in the sense of Definition 8.2.1), including AC^0, TC^0_2, NC^1, $\mathsf{P/poly}$, etc.

We say that a deterministic algorithm is *nontrivial* if it runs in time $2^n/n^{\omega(1)}$. We may use this terminology to talk about nondeterministic and randomized algorithms with similar time bounds.

The following folklore result shows that to prove a circuit lower bound for P it is enough to obtain a circuit lower bound for the non-uniform class $\mathsf{P/poly}$.

Lemma 8.2.2. *Let $\mathcal{C}_d[\mathrm{poly}(n)]$ be a reasonable circuit class. If $\mathsf{P} \subseteq \mathcal{C}_d[\mathrm{poly}(n)]$, then for every $b \in \mathbb{N}$ there exists a $t \in \mathbb{N}$ such that every Boolean circuit over n inputs of size n^b admits an equivalent circuit from $\mathcal{C}_d$ of size n^t.*

Proof. Assume that $\mathsf{P} \subseteq \mathcal{C}_d[\mathrm{poly}(n)]$. Consider the following problem:

$$\mathsf{Circuit\text{-}Eval}_b = \{\langle E, x\rangle \ : \ E \text{ is a circuit on } n \text{ variables of size } \leq n^b \text{ and } E(x) = 1\}$$

Clearly, $\mathsf{Circuit\text{-}Eval}_b$ is in P (for any fixed b), and thus there exists t such that $\mathsf{Circuit\text{-}Eval}_b \in \mathcal{C}_d[n^t]$. In other words, there exists a sequence $\{D_n\}_{n\in\mathbb{N}}$ of circuits from $\mathcal{C}_d$ of size $O(n^t)$ that computes $\mathsf{Circuit\text{-}Eval}_b$.

Let $E_n : \{0,1\}^n \to \{0,1\}$ be a function over n Boolean variables computed by a circuit of size at most n^b. We can hardwire the description of E_n inside circuit D_n (recall that

[11] This allows us to hardwire some values without increasing the depth of the circuit.

$\mathcal{C}$ is reasonable, and that this operation does not increase the depth of the circuit). The resulting circuit is in $\mathcal{C}_d$, has size at most n^t, and it computes E_n by definition of D_n. □

The next definition will play an important role in many results discussed later.

Definition 8.2.3 (Properties that are useful against $\mathcal{C}$ [201])**.** *A property of Boolean functions is a subset of the set of all Boolean functions. For a typical circuit class $\mathcal{C}$, a property $\mathcal{P}$ is said to be* useful *against $\mathcal{C}$ if, for all k, there are infinitely many positive integers n such that*

- *$\mathcal{P}(f_n)$ is true for at least one function $f_n : \{0,1\}^n \to \{0,1\}$, and*
- *$\mathcal{P}(g_n)$ is false for all functions $g_n : \{0,1\}^n \to \{0,1\}$ that admit circuits from $\mathcal{C}[n^k]$.*

We say that $\mathcal{P}$ is a Γ-property *if, given the truth-table $tt(f_n) \in \{0,1\}^N$ (where $N = 2^n$) of any Boolean function $f_n : \{0,1\}^n \to \{0,1\}$, $\mathcal{P}(f_n)$ can be decided in complexity class Γ. In other words, the language*

$$L_{\mathcal{P}} = \{w \in \{0,1\}^N \mid w = tt(f_n) \text{ for some function } f_n : \{0,1\}^n \to \{0,1\} \text{ with } \mathcal{P}(f_n) = 1\}$$

is in Γ.

A useful property distinguishes some "hard" function from all easy ones. This is weaker than the notion of *natural properties* studied by [154], which also requires $\mathcal{P}$ to be dense, i.e., $\mathcal{P}(f) = 1$ for a non-negligible fraction of functions.

Recall that a verifier V for a language $L \in \mathsf{NTIME}[t(n)]$ satisfies the following properties:

- $V(x, w)$ runs in deterministic time $O(t(n))$, where $n = |x|$.
- $x \in L$ if and only if there exists $w \in \{0,1\}^{O(t(n))}$ such that $V(x, w) = 1$.

If $L \in \mathsf{NEXP}$ and V is a verifier for L running in time $2^{n^{O(1)}}$, we say that V is a NEXP-verifier for L. Similarly, we may talk about NE-verifiers running in time $2^{O(n)}$.

Definition 8.2.4. *Let $\mathcal{C}$ be a typical circuit class. We say that a NEXP-verifier V for a language $L \in \mathsf{NEXP}$* admits witness circuits from $\mathcal{C}[s(n)]$ *if for all $x \in L$, there exists a circuit $C \in \mathcal{C}[s(n)]$ such that $V(x, tt(C)) = 1$.*

Proposition 8.2.5 (Impagliazzo et al. [104], Williams [199])**.** *Let $\mathcal{C}$ be a typical circuit class. If $\mathsf{NEXP} \subset \mathcal{C}$ then for any language $L \in \mathsf{NEXP}$ and every NEXP-verifier V for L, there exists $c \in \mathbb{N}$ such that V admits witness circuits from $\mathcal{C}[n^c]$.*

Definition 8.2.6. *Given functions $f, g : \{0,1\}^n \to \{0,1\}$ and $\delta > 0$, we say that g computes f with advantage δ if*

$$\Pr_{x \in_R \{0,1\}^n}[f(x) = g(x)] \geq \frac{1}{2} + \delta.$$

The results for learning algorithms and lossy compression rely on the following fact.

Lemma 8.2.7 ("Random functions are hard to approximate")**.**
There exists a constant $\alpha > 0$ such that for any sufficiently large n, there exists a function $h : \{0,1\}^n \to \{0,1\}$ that cannot be computed with advantage $\delta > 0$ by any circuit of size $\alpha \cdot 2^n \delta^2 / n$.

Proof. Fix any circuit $C : \{0,1\}^n \to \{0,1\}$. Using the Chernoff-Hoeffding bound, we get that the probability that C computes a random function $r : \{0,1\}^n \to \{0,1\}$ with advantage δ is at most $\exp(-2\delta^2 N)$, where $N = 2^n$ as usual. There are at most $2^{O(s(n) \log s(n))}$ functions on n inputs computed by circuits with $s(n)$ gates. Therefore, it follows by a simple union bound that for some $\alpha > 0$, there exists a function h that is not computed with advantage δ by any circuit of size $\alpha \cdot 2^n \delta^2 / n$. □

8.3 Lower bounds from non-trivial satisfiability algorithms

In this section we present the transference theorem for satisfiability algorithms. We start with the following definition.

Definition 8.3.1. *Let $\mathcal{C}$ be a circuit class. We define the computational problem* Equiv-AND-$\mathcal{C}$ *as follows. Given the description of circuits from $\mathcal{C}$ computing functions $f_1, f_2, f_3 : \{0,1\}^n \to \{0,1\}$, check if $\mathsf{AND}(f_1, f_2)(x) = f_3(x)$ for every $x \in \{0,1\}^n$. The* Equiv-OR-$\mathcal{C}$ *problem is defined analogously.*

Remark 8. *Observe that if $\mathcal{C}_d$ is reasonable, then an algorithm for* Equiv-AND-$\mathcal{C}_d$ *can be used to solve $\mathcal{C}_d$-SAT. Moreover, the same algorithm can be used to solve* Equiv-OR-$\mathcal{C}$*, since*

two functions are equivalent if and only if their negations are equivalent (by assumption, any reasonable circuit class is closed under negations).

The proof presented here follows closely the original argument used by Williams [198], which works for $\mathsf{P/poly}$. However, we introduce a new technique that allows us to obtain an equivalent $\mathcal{C}$-circuit from a general $\mathsf{P/poly}$-circuit. It simplifies the proof in [199], and provides a tighter connection between satisfiability algorithms and circuit lower bounds in the case of bounded-depth circuits. The proof of the next lemma is partially inspired by some ideas in Rossman [160].

Lemma 8.3.2 ("Conversion Lemma")**.** *Let $\mathcal{C}$ be a reasonable circuit class, and suppose that $\mathsf{P} \subseteq \mathcal{C}$. In addition, assume that there is a nontrivial algorithm for $\mathsf{Equiv\text{-}AND\text{-}}\mathcal{C}$. Then there exists a nondeterministic algorithm $\mathcal{N}$ with the following properties. Given as input any circuit B over m variables of size m^b,*

- *$\mathcal{N}$ has at least one accepting path, and in every accepting path it outputs a circuit G from $\mathcal{C}[m^t]$ that is equivalent to B (where $t = O(b)$).*
- *$\mathcal{N}$ runs in time at most $\frac{2^m}{s(m)}$, for some superpolynomial function $s(m)$.*

Proof. Let $\mathcal{A}$ be an algorithm for $\mathsf{Equiv\text{-}AND\text{-}}\mathcal{C}$ running in time $2^m/a(m)$, for a superpolynomial function $a(m)$. We proceed as follows. Let $x_1, x_2, \ldots, x_m, g_1, \ldots, g_k$ for $k = m^b$ be a topological sort of the gates of B, where each gate $g_i \in \{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$ has fan-in at most two. We will guess and verify (by induction) equivalent $\mathcal{C}$-circuits of size m^t for each gate g_i in B. Since $\mathsf{P} \subseteq \mathcal{C}$, it follows from Lemma 8.2.2 that the functions computed at the internal gates of B admit such circuit.

More details follow. Suppose (by induction) that $\mathcal{N}$ has produced equivalent $\mathcal{C}$-circuits $B_i^{\mathcal{C}}$ of size at most m^t for every gate g_i of B, where $i < \ell$ (otherwise it has aborted already). If g_ℓ is an AND gate with inputs g_{i_1}, g_{i_2}, where $i_1, i_2 < \ell$, $\mathcal{N}$ guesses a circuit $B_\ell^{\mathcal{C}}$ in $\mathcal{C}[m^t]$ over the same input variables, then use the $\mathsf{Equiv\text{-}AND\text{-}}\mathcal{C}$ algorithm to check if $\mathsf{AND}(B_{i_1}^{\mathcal{C}}, B_{i_2}^{\mathcal{C}})$ and $B_\ell^{\mathcal{C}}$ are equivalent. $\mathcal{N}$ rejects if these circuits are not equivalent, otherwise it continues the computation, completing the induction step. If g_ℓ corresponds to an OR gate, a similar computation is performed, this time applying an algorithm for $\mathsf{Equiv\text{-}OR\text{-}}\mathcal{C}$ (check Remark

8). Finally, if g_ℓ is a NOT gate, using the fact that $\mathcal{C}$ is reasonable, it is possible to produce in polynomial time an equivalent $\mathcal{C}$-circuit for g_ℓ of the same size. This completes the induction step. Observe that the base case is trivial.

Note that $\mathcal{N}$ runs algorithm $\mathcal{A}$ for at most k times, i.e., a polynomial number of times. In addition, each execution is performed over circuits from $\mathcal{C}$ of size $O(m^t)$. Therefore the total running time of $\mathcal{N}$ is $\text{poly}(m) \cdot 2^m/a(m)$, for some superpolynomial function $a(m)$. Setting $s(m) = a(m)/\text{poly}(m)$ completes the proof of Lemma 8.3.2. □

In addition, we will need the following auxiliary results, whose notation we borrow from Williams [199].

Definition 8.3.3. *The computational problem* Succinct-SAT *is defined as follows. Given a circuit C over n input variables, denote by F_C the instance of* 3-SAT *obtained by evaluating C over all inputs in lexicographic order (i.e., F_C is the 2^n-bit string representing the truth-table $tt(C)$ of C). Decide if F_C is satisfiable.*

We say that F_C is the *decompression* of C, and call C the *compression* of F_C.

Lemma 8.3.4 (Tourlakis [186], Fortnow et a. [72], Williams [198])**.** *There is a fixed constant $c > 0$ for which the following holds. For every $L \in \text{NTIME}[2^n]$ there is a polynomial time reduction from L to* Succinct-SAT *that maps every input x of size n to a circuit C_x over at most $n + c \log n$ input variables and size $O(n^c)$, such that $x \in L$ if and only if the decompressed formula F_{C_x} is satisfiable (observe that this is a formula of size $2^n \text{poly}(n)$).*

Definition 8.3.5. *We say that* Succinct-SAT *admits* succinct satisfying assignments *if there exists a constant $c > 0$ such that for every language $L \in \text{NTIME}[2^n]$ the following holds. Given any $x \in L$, there exists some circuit W_x of polynomial size over $k \leq n + c \log n$ input variables for which the assignment $z_i = W(i)$ for $i \in \{1, \ldots, 2^k\}$ is a satisfying assignment for F_{C_x}, where C_x is the circuit obtained from the reduction to* Succinct-SAT *given by Lemma 8.3.4.*

The following lemma is an easy consequence of Proposition 8.2.5.

Lemma 8.3.6. *If* NEXP $\subseteq$ P/poly *then* Succinct-SAT *admits succinct satisfying assignments.*

We use these auxiliary results to prove the following proposition. For simplicity, we only state it for polynomial size classes, but a parameterized version can be obtained using the same techniques.

Proposition 8.3.7. *Let $\mathcal{C} = \mathcal{C}_d[\text{poly}(n)]$ be a reasonable circuit class. If there exist a nontrivial algorithm for* Equiv-AND-$\mathcal{C}$*, then* NEXP $\not\subseteq \mathcal{C}$*.*

Proof. Let $\mathcal{A}$ be a nontrivial algorithm for Equiv-AND-$\mathcal{C}$, and assume for the sake of a contradiction that NEXP $\subseteq \mathcal{C}$. We use these assumptions to show that every language $L \in \text{NTIME}[2^n]$ is in $\text{NTIME}[o(2^n)]$, a contradiction to the nondeterministic time hierarchy theorem ([52], [171], [206]).

The proof relies on the fact that every language $L \in \text{NTIME}[2^n]$ can be efficiently reduced to an instance of the Succinct-SAT problem (Lemma 8.3.4). In other words, there is a polynomial time algorithm that maps any input $x \in \{0,1\}^n$ to a circuit D_x on $n + c \log n$ input variables and at most $O(n^c)$ gates such that $x \in L$ if and only if the decompression $F_x = tt(D_x)$ of D_x is satisfiable.

It follows from NEXP $\subseteq \mathcal{C} \subseteq$ P/poly ($\mathcal{C}$ is reasonable) and Lemma 8.3.6 that if F_x is satisfiable then there is a satisfying assignment encoded by a circuit E_x of polynomial size over $n + O(\log n)$ variables. Summarizing what we have so far:

$$x \in L \iff \exists \text{ circuit } E : \{0,1\}^{n+O(\log n)} \to \{0,1\} \text{ of size } O(n^d) \text{ such that } F_x(tt(E)) = 1,$$

where $F_x = tt(D_x)$ is a 3-CNF formula and $D_x : \{0,1\}^{n+O(\log n)} \to \{0,1\}$ is an arbitrary circuit (not necessarily in $\mathcal{C}$) of size $O(n^c)$ encoding this formula.

Our nontrivial algorithm for L now guesses a candidate circuit E of this form. It uses D_x and three copies of E to build a circuit $B = B(D_x, E)$ of size $O(n^b)$ over $n + O(\log n)$ inputs such that:

$$B \text{ is satisfiable} \iff \text{some clause } C_i \text{ of } F_x \text{ is not satisfiable by the assignment } tt(E).$$

The description of B is as follows. An input y to B is interpreted as an integer i, and B uses this index to obtain from D_x the description of the i-th clause C_i in F_x. Let z_1, z_2, z_3 be the literals in C_i. Circuit B uses three copies of E to obtain the Boolean values of the variables corresponding to these literals, and finally outputs 1 if and only if these values

do not satisfy C_i. This last verification can be done by a polynomial size circuit. Observe that B is not a $\mathcal{C}$-circuit: D_x and E are arbitrary circuits, these circuits are composed, and there is additional circuitry computing the final output value of B. Overall, we obtain:

$$x \in L \iff \text{circuit } B : \{0,1\}^{n+O(\log n)} \to \{0,1\} \text{ is unsatisfiable.} \tag{8.1}$$

Note that all these steps can be performed in $\mathsf{NTIME}[\mathrm{poly}(n)]$. Recall that we can use $\mathsf{Equiv\text{-}AND\text{-}}\mathcal{C}$ to solve $\mathcal{C}\text{-}\mathsf{SAT}$ in less than 2^n steps, but B is not a circuit from $\mathcal{C}$. We can assume without loss of generality that B is a circuit of size m^b (where $m = n + O(\log n)$) consisting of AND, OR and NOT gates of fan-in at most two.

While in Williams' original proof there is a step that guesses and verifies an equivalent $\mathcal{C}$-circuit for D_x (and already assumes E in $\mathcal{C}[\mathrm{poly}(n)]$ with some extra work), our nondeterministic algorithm for L produces directly an equivalent $\mathcal{C}$-circuit for the final circuit B. Under our assumptions, Lemma 8.3.2 can be applied, and it allows the nondeterministic algorithm for L to obtain a circuit G over m inputs from $\mathcal{C}[m^t]$ that is equivalent to B. This step can be performed in time $2^m/s(m)$ for a superpolynomial function $s(m)$. Since $m = n + O(\log n)$, this running time is still nontrivial in n.

Using condition (8.1), it follows that $x \in L$ if and only if G is unsatisfiable. Finally, since $\mathcal{C}$ is reasonable, we can use algorithm $\mathcal{A}$ to check if this is true, in which case our algorithm for L accepts input x. Again, this is a computation that can be performed in nontrivial running time by our assumption over $\mathcal{A}$. Overall, it follows that we can decide L in $\mathsf{NTIME}[o(2^n)]$, which completes the proof of the theorem. □

Corollary 8.3.8. *Let $\mathcal{C} = \mathcal{C}_d[\mathrm{poly}(n)]$ be a reasonable circuit class. If there exist nontrivial satisfiability algorithms for both $\mathsf{AND}[3] \circ \mathcal{C}$ and $\mathsf{AND}[2] \circ \mathsf{OR}[2] \circ \mathcal{C}$, then $\mathsf{NEXP} \not\subseteq \mathcal{C}$.*

Proof. It is enough to observe that these satisfiability algorithms can be used to solve $\mathsf{Equiv\text{-}AND\text{-}}\mathcal{C}$ in nontrivial running time (Proposition 8.3.7). Let f_1, f_2, f_3 be functions from $\mathcal{C}$. Then

$$\neg\mathsf{EQUIV}(\mathsf{AND}(f_1, f_2), f_3) \iff \mathsf{XOR}(\mathsf{AND}(f_1, f_2), f_3) \text{ is satisfiable.} \tag{8.2}$$

For bits $a, b \in \{0,1\}$, we have $\mathsf{XOR}(a,b) \equiv \mathsf{OR}(\mathsf{AND}(a, \bar{b}), \mathsf{AND}(\bar{a}, b))$. Using de Morgan's rules and combining gates, it is not hard to see that

$$\mathsf{XOR}(\mathsf{AND}(f_1, f_2), f_3) \equiv \mathsf{OR}(\mathsf{AND}(f_1, f_2, \bar{f_3}), \mathsf{AND}(\mathsf{OR}(\bar{f_1}, \bar{f_2}), f_3)).$$

It follows from (8.2) that an algorithm for ¬Equiv-AND-$\mathcal{C}$ should output 1 if and only if either $\mathsf{AND}(f_1, f_2, \bar{f}_3)$ or $\mathsf{AND}(\mathsf{OR}(\bar{f}_1, \bar{f}_2), f_3)$ is satisfiable. Since $\mathcal{C}$ is reasonable, a circuit for $\bar{f}_i$ can be computed efficiently from a circuit for f_i. Hence nontrivial algorithms for $\mathsf{AND}[3] \circ \mathcal{C}$-SAT and $\mathsf{AND}[2] \circ \mathsf{OR}[2] \circ \mathcal{C}$-SAT can be used to solve ¬Equiv-AND-$\mathcal{C}$, which completes the proof. □

8.3.1 A remark for the algorithm designer

It is hard to find satisfiability algorithm for expressive circuit classes even when we allow very modest running times, such as $2^n/n^{\log n}$. Here we mention a weaker assumption on the algorithmic side that may be relevant when proving lower bounds.[12]

Definition 8.3.9 ("Algorithms useful for circuit lower bounds")**.**
Let $\mathcal{C}$ be a circuit class. A nondeterministic algorithm $\mathcal{A}$ for Equiv-AND-$\mathcal{C}$ *is* useful *if the following conditions hold:*

- *Every path of the (nondeterministic) computation of $\mathcal{A}$ either outputs "abort", or provides the correct answer.*
- *At least one path of the computation of $\mathcal{A}$ does not abort, and runs in time bounded by $2^n/s(n)$ for some superpolynomial function $s(n)$.*

Proposition 8.3.10. *Let $\mathcal{C} = \mathcal{C}_d[\mathrm{poly}(n)]$ be a reasonable circuit class. If there exists a useful algorithm for* Equiv-AND-$\mathcal{C}$ *then* $\mathsf{NEXP} \not\subseteq \mathcal{C}$.

Proof. Observe that the proof of Proposition 8.3.7 still holds with such algorithms, provided that we abort in any computation path that runs for more than $2^n/s(n)$ steps. It is still the case that $x \in L$ if and only if there exists a computation path that accepts x. More precisely, if $x \notin L$, even if equivalent circuits are guessed and verified in each stage, a useful algorithm for Equiv-AND-$\mathcal{C}$ will never output "yes" in the last step of the computation that checks if the final circuits is equivalent to the zero function (i.e., it is unsatisfiable). On the other hand, for $x \in L$, it is clear from the definition of useful algorithm that some computation path will accept in nontrivial running time. □

[12] This is not the most encompassing definition, but it is a fairly natural one.

Observe that useful algorithms for *unsatisfiability* also lead to circuit lower bounds, since these can be used in place of an Equiv-AND-$\mathcal{C}$ algorithm. The same is true for satisfiability algorithms, since useful algorithms are closed under complementation. In Section 8.5.2 we will use Proposition 8.3.10 to prove that derandomization implies circuit lower bounds (Proposition 8.1.6).

Why is this a natural relaxation? Suppose there exists a class $\mathcal{C}$ such that for any circuit D in this class, there exists some subset $S \subset [n]$ of the inputs of D such that by trying all assignments to the variables in S, we can check on *average time* strictly less than $2^{n-|S|}$ (over the restrictions) the satifiability of the remaining circuits. Then $\mathcal{C}$ admits a useful satisfiability algorithm, since the set S can be guessed at the beginning of the execution. For the reader familiar with the satisfiability algorithm for small threshold circuits described by Impagliazzo, Paturi and Schneider [107], it means that their algorithm gives more than what is needed for lower bounds. There the expected running time is nontrivial over the subset of inputs to be restricted, which is a stronger guarantee. It is sufficient that a single subset provides a nontrivial running time.

8.4 Useful properties and circuit lower bounds

In this section we focus on the relation between useful properties and circuit lower bounds, a connection investigated in a recent paper written by Williams [201]. Recall that an algorithm that computes a property of Boolean functions receives as input a string of size $N = 2^n$ representing the truth-table $tt(f)$ of a function $f : \{0,1\}^n \to \{0,1\}$. We start with the following simple result.

Proposition 8.4.1 ("Useful NP-property yields useful P-property")**.**
Let $\mathcal{C}$ be a typical circuit class, and let $s : \mathbb{N} \to \mathbb{N}$ be any function. If there is a $\mathsf{NP}/s(N)$-property useful against $\mathcal{C}$ then there is a $\mathsf{P}/s(N)$-property useful against $\mathcal{C}$.

Proof. First we prove the proposition without advice, then we observe that the same proof works in the presence of advice strings as well. Let $\mathcal{P}$ be a NP-useful property against $\mathcal{C}$. In other words, for any fixed k, there exists an infinite subset $S_k \subseteq \mathbb{N}$ such that for any $n \in S_k$:

- $\mathcal{P}(f_n) = 1$ for at least one function $f_n : \{0,1\}^n \to \{0,1\}$.
- $\mathcal{P}(g_n) = 0$ for any function $g_n : \{0,1\}^n \to \{0,1\}$ computed by circuits in $\mathcal{C}[n^k]$.

In addition, there exists a polynomial time verifier $V_{\mathcal{P}} : \{0,1\}^N \times \{0,1\}^{N^c - N} \to \{0,1\}$ (where $N = 2^n$ and $c \in \mathbb{N}$) for $L_{\mathcal{P}}$. Put another way, for any function h_n,

$$\mathcal{P}(h_n) = 1 \qquad \Longleftrightarrow \qquad \exists w \in \{0,1\}^{N^c - N} \text{ such that } \; V_{\mathcal{P}}(tt(h_n), w) = 1.$$

Let $A = \{n' \mid n' = cn, n \in \mathbb{N}\}$. For convenience, set $N' = 2^{n'} = N + (N^c - N)$. We define a predicate $\mathcal{P}'$ defined on any function over n' inputs, where $n' \in A$ (the definition of $\mathcal{P}'$ over functions with a different number of inputs can be arbitrary). For any $h'_{n'} : \{0,1\}^{n'} \to \{0,1\}$, view its representation $tt(h'_{n'}) \in \{0,1\}^{N'}$ as a pair of strings $(tt(h_n), w)$, where $tt(h_n) \in \{0,1\}^N$ and $w \in \{0,1\}^{N^c - N}$. To be more precise, let $h_n : \{0,1\}^n \to \{0,1\}$ be the restriction of $h'_{n'}$ defined by $h_n(x) = h'_{n'}(x0^{(c-1)n})$, where $x \in \{0,1\}^n$. Finally, let

$$\mathcal{P}'(h'_{n'}) = 1 \qquad \Longleftrightarrow \qquad V_{\mathcal{P}}(tt(h_n), w) = 1.$$

We claim that $\mathcal{P}'$ is a P-property that is useful against $\mathcal{C}$. First observe that since $V_{\mathcal{P}}$ is an efficient algorithm, $\mathcal{P}'$ can be computed in time polynomial in $N' = |tt(h'_{n'})|$. Fix any $k \in \mathbb{N}$. We need to define an infinite set $S'_k \subseteq A$ such that for every $n' \in S'_k$,

- $\mathcal{P}(f'_{n'}) = 1$ for at least one function $f'_{n'} : \{0,1\}^{n'} \to \{0,1\}$.
- $\mathcal{P}(g'_{n'}) = 0$ for any function $g'_{n'} : \{0,1\}^{n'} \to \{0,1\}$ computed by circuits in $\mathcal{C}[n'^k]$.

Let $S'_k = \{n' \mid n' = cn, n \in S_{k+1}\}$. This set is infinite because so is S_{k+1}. Let $n' \in S'_k$. It follows from the definition of S_{k+1} that there is a function $f_n : \{0,1\}^n \to \{0,1\}$ for which $\mathcal{P}(f_n) = 1$. Hence there exists $w \in \{0,1\}^{N^c - N}$ such that $V_{\mathcal{P}}(tt(f_n), w) = 1$. By construction, the corresponding function $f'_{n'} : \{0,1\}^{n'} \to \{0,1\}$ whose truth-table is the concatenation of the pair (f_n, w) satisfies $\mathcal{P}'$.

Finally, in order to establish the second bullet, assume for the sake of a contradiction that there exists a function $g'_{n'} : \{0,1\}^{n'} \to \{0,1\}$ computed by circuits from $\mathcal{C}[n'^k]$ for which $\mathcal{P}'(g'_{n'}) = 1$. Clearly, the function $g_n : \{0,1\}^n \to \{0,1\}$ defined as before by the restriction $g_n(x) = g'_{n'}(x0^{(c-1)n})$ also admits circuits from $\mathcal{C}$ of size $n'^k = (cn)^k \leq n^{k+1}$, for

sufficiently large values of n. But then $\mathcal{P}(g_n) = 0$, since $n \in S_{k+1}$. However, this contradicts the assumption that $\mathcal{P}'(g'_{n'}) = 1$, since in this case there is no $w \in \{0,1\}^{N^c - N}$ such that $V_{\mathcal{P}}(tt(g_n), w) = 1$. In other words, for every function $g'_{n'}$ with $n' \in S'_k$ that is computed by circuits from $\mathcal{C}[n'^k]$, we have $\mathcal{P}'(g'_{n'}) = 0$.

If the original verifier works with advice strings of length $s(N)$, then property $\mathcal{P}'$ can be decided correctly using the same advice. However, the definition of $\mathcal{P}'$ over functions on $n' = cn$ inputs is based on the definition of $\mathcal{P}$ over functions on n inputs. Therefore, the advice for the new algorithm is of size $s(N'^{1/c})$, since it gets as input truth-tables of size $N' = N^c$. Assuming that $s(.)$ is non-decreasing and $c \geq 1$, it follows that $\mathcal{P}'$ can be decided with advice of size $s(N'^{1/c}) \leq s(N')$. This completes the proof of Proposition 8.4.1. □

The new useful property may not be dense, even if the original property is dense. The reason is that there may be just a few certificates for each hard function, thus almost no function will satisfy the newly defined property. However, if we start with an RP-natural property useful against $\mathcal{C}$ (i.e., a dense property in which every hard function has many certificates), the proof of Proposition 8.4.1 yields a corresponding P-natural property.

The next proposition clarifies the relation between NEXP circuit lower bounds and the existence of properties that are useful against $\mathcal{C}$. Recall that for any typical circuit class, standard arguments can be used to prove that $\mathsf{NEXP} \not\subseteq \mathcal{C}$ if and only if $\mathsf{NE} \not\subseteq \mathcal{C}$.

Proposition 8.4.2. *Let $\mathcal{C}$ be a typical class. Then $\mathsf{NEXP} \not\subseteq \mathcal{C}$ if and only if there exists a $\mathsf{P}/\log N$-property that is useful against $\mathcal{C}$.*

Proof. Let $N = 2^n$ as usual. First assume that $\mathsf{NEXP} \not\subseteq \mathcal{C}$, and let $L \in \mathsf{NE}\backslash\mathcal{C}$. Let $L' = L \cup \{1^n \mid n \in \mathbb{N}\}$, and notice that $L' \in \mathsf{NE}\backslash\mathcal{C}$. For every $n \in \mathbb{N}$, let $b(n)$ be the number of strings of size n in L'. Observe that $b(n) \in [1, 2^n]$. Therefore $b(n)$ can be encoded by a string $a(n) \in \{0,1\}^{\log N}$. Let $f_n = L_n$, i.e., $f_n(x) = 1$ if and only if $x \in L$. Consider the property $\mathcal{P}$ such that $\mathcal{P}(g) = 1$ if and only if $g = f_n$ for some $n \in \mathbb{N}$. We claim that $\mathcal{P}$ is a $\mathsf{NP}/\log N$-property that is useful against $\mathcal{C}$. Let V' be an NE-verifier for L' accepting witnesses of size 2^{cn}.

Clearly, $\mathcal{P}$ is useful against $\mathcal{C}$, because $L' \notin \mathcal{C}$. On the other hand, the following NP-verifier decides $\mathcal{P}$ when it is given the correct advice string $a(n)$:

Verifier V for $\mathcal{P}$:
On inputs $tt(h) \in \{0,1\}^N$ and advice string $z \in \{0,1\}^{\log N}$, reject if $|h^{-1}(1)| \neq z$. Otherwise, guess witnesses $w_x \in \{0,1\}^{N^c}$ for every $x \in h^{-1}(1)$, and accept if and only if $V'(x, w_x) = 1$ for every such x.

Clearly, when $z = a(n)$, the only function over n inputs accepted by V is $f_n = L_n$. In addition, V runs in time $\mathsf{poly}(N)$. It follows that $\mathcal{P}$ is computed in $\mathsf{NP}/\log N$. Therefore, there is a $\mathsf{NP}/\log N$-property that is useful against $\mathcal{C}$, and Proposition 8.4.1 guarantees the existence of a $\mathsf{P}/\log N$-property useful against $\mathcal{C}$.

Now suppose that there exists a $\mathsf{P}/\log N$-property $\mathcal{P}'$ that is useful against $\mathcal{C}$. We use this assumption to define a NEXP-verifier V' that does not admit witness circuits of polynomial size. Observe that it follows then from Proposition 8.2.5 that $\mathsf{NEXP} \not\subseteq \mathcal{C}$, which completes the proof our result.

Let $\mathcal{A}'$ be an algorithm running in time N^d that decides $\mathcal{P}'$ on inputs $tt(f) \in \{0,1\}^N$ when it is given access to an appropriate advice string $a(N) \in \{0,1\}^{\log N}$, i.e, a string of size n. Consider the following verifier.

NEXP-verifier V':
On input $\langle x, w \rangle$, where $x \in \{0,1\}^n$ and $w \in \{0,1\}^N$, output $\mathcal{A}'(w)/x$ (i.e., run $\mathcal{A}'$ on input w with advice string x).

First observe that V' is a NEXP-verifier. Fix any $c \in \mathbb{N}$. We prove that V' does not admit witness circuits from $\mathcal{C}[n^c]$. First, there are infinitely many inputs n for which $\mathcal{P}'$ correctly discriminates a hard function $h_n : \{0,1\}^n \to \{0,1\}$ from a function in $\mathcal{C}[n^c]$. For any such value of n, there is a correct advice string $a(N)$ for which algorithm $\mathcal{A}'$ computes $\mathcal{P}'$. However, whenever $x = a(N)$, it follows from the definition of V' that it only accepts certificates for x that do not correspond to any truth-table from $\mathcal{C}[n^c]$. In addition, V' accepts at least one truth-table, by definition of $\mathcal{P}'$. As discussed before, this completes the proof of Proposition 8.4.2. □

One may be tempted to pose the following conjecture.

Conjecture 8.4.3. *Let $\mathcal{C}$ be a typical circuit class. If there exists a $\mathsf{P}/O(\log N)$-property that is useful against $\mathcal{C}$, then there is a P-property that is useful against $\mathcal{C}$.*

We will see shortly that if a slightly more general version of this conjecture holds, then a generic NEXP circuit lower bound can always be converted into a $\mathsf{NE} \cap \mathsf{coNE}$ lower bound, a rather surprising consequence, given its generality.

Now we move to the relation between useful properties decided without advice and circuit lower bounds.

Proposition 8.4.4. *For any typical $\mathcal{C}$, if $\mathsf{NE} \cap \mathsf{coNE} \not\subseteq \mathcal{C}$ then there exists a P-property that is useful against $\mathcal{C}$.*

Proof. Let $L \in \mathsf{NE} \cap \mathsf{coNE} \backslash \mathcal{C}$, and let V^0 and V^1 be verifiers running in time $2^{O(n)}$ for $n = |x|$ such that:

$$x \in L \iff \exists w_x \in \{0,1\}^{2^{O(n)}} \text{ such that } V^1(x, w_x) = 1.$$
$$x \notin L \iff \exists w_x \in \{0,1\}^{2^{O(n)}} \text{ such that } V^0(x, w_x) = 1.$$

We view L as a family of functions $f = \{f_n\}_{n \in \mathbb{N}}$, where $f_n^{-1}(1) = L_n$. Let $\mathcal{P} = \{f_n \mid n \in \mathbb{N}\}$. First observe that this property is useful against $\mathcal{C}$, since $L \notin \mathcal{C}$. In addition, there is an efficient verifier $V_{\mathcal{P}}$ for $\mathcal{P}$: on input a string $tt(h) \in \{0,1\}^N$ representing the truth-table of a function $h : \{0,1\}^n \to \{0,1\}$, guess 2^n certificates $y_x \in \{0,1\}^{N^c}$, one for each $x \in \{0,1\}^n$, and accept if and only if $V^{h(x)}(x, y_x) = 1$ for every such x. Clearly, $V_{\mathcal{P}}$ is a NP-verifier for $\mathcal{P}$. It follows then from Proposition 8.4.1 that there exists a P-property $\mathcal{P}'$ that is useful against $\mathcal{C}$, which completes the proof. □

Conversely, which consequences can we obtain from the existence of P-properties (without advice) that are useful against $\mathcal{C}$? The following result is implicit in the work of Williams [201], and shows that without advice even stronger consequences can be obtained (although in the quasipolynomial size regime).

Proposition 8.4.5. *Let $\mathcal{C}$ be a typical circuit class. If for every $c \in \mathbb{N}$ there exists a P-property that is useful against $\mathcal{C}[n^{\log^c n}]$, then $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathcal{C}[n^{\log n}]$.*

We give a self-contained proof of this result in Section 8.7.

Proposition 8.4.5 sheds some light into Conjecture 8.4.3. It shows that if the analogue of this conjecture for quasipolynomial size circuits holds, then NEXP lower bounds against such circuits can be translated into similar $\mathsf{NE} \cap \mathsf{coNE}$ circuit lower bounds (via a generalization of Proposition 8.4.2 to quasipolynomial size circuits).

Given the statement of Propositions 8.4.4 and 8.4.5, it is plausible to conjecture that there is a tight correspondence between useful properties computed without advice and circuit lower bounds for $\mathsf{NE} \cap \mathsf{coNE}$.

Conjecture 8.4.6. *Let $\mathcal{C} = \mathcal{C}[\mathrm{poly}]$ be a typical circuit class. Then $\mathsf{NE} \cap \mathsf{coNE} \not\subseteq \mathcal{C}$ if and only if there exists a P-property that is useful against $\mathcal{C}$.*

We will see in Section 8.5 that useful properties are powerful enough to simplify and generalize many results of the form "nontrivial algorithms yield circuit lower bounds". In particular, a proof of Conjecture 8.4.6 would provide stronger transference theorems in different frameworks.

8.4.1 Satisfiability algorithms and useful properties

It is possible to formulate the main result from Section 8.3 as follows: the existence of nontrivial satisfiability algorithms leads to useful properties, which in turn imply circuit lower bounds. This can be accomplished using the fact that the nondeterministic hierarchy theorem also holds for unary languages. In other words, if there exists a nontrivial SAT algorithm for a circuit class $\mathcal{C}$, the proof of Proposition 8.3.7 shows that any verifier for a hard unary language must have infinitely many inputs that only admit certificates of high $\mathcal{C}$-circuit complexity. This verifier can be used to define a property that is useful against $\mathcal{C}$: given a truth table $tt(h_n)$, check if it is a valid certificate for the input 1^n.

More specifically, satisfiability algorithms for polynomial size circuits lead to P-properties useful against circuits of polynomial size, while algorithms for quasipolynomial size circuits lead to P-properties useful against circuits of such size. The reader should compare the transference theorems from [199] and [201] (Propositions 8.1.1 and 8.1.3, respectively) to the statements of Propositions 8.4.2 and 8.4.5. If Conjecture 8.4.6 is true, the existence of

nontrivial satisfiability algorithms for $\mathcal{C}[\text{poly}]$ would imply that $\mathsf{NE} \cap \mathsf{coNE} \not\subseteq \mathcal{C}[\text{poly}]$, a new result.

8.5 Applications and additional connections

8.5.1 Lower bounds from lossy compression

In this section we prove the transference theorem obtained by Chen et al. [49], which we state again for convenience.

Proposition 8.5.1 (Compression yields circuit lower bounds [49])**.**
Let $\mathcal{C}$ be a typical circuit class. Suppose that for every $c \in \mathbb{N}$ there is a deterministic polynomial-time algorithm that compresses a given truth table of an n-variate Boolean function $f \in \mathcal{C}[n^c]$ to an equivalent circuit of size $o(2^n/n)$. Then $\mathsf{NEXP} \not\subseteq \mathcal{C}$.

As mentioned before, it is possible to show a similar result from the existence of lossy compression algorithms.

Definition 8.5.2 (Lossy compression scheme)**.** *Let $\mathcal{C}$ be a typical circuit class. We say that a* deterministic *algorithm $\mathcal{A}$ is a* $(\delta(n), s(n))$-compression algorithm *for $\mathcal{C}$ if $\mathcal{A}$ runs in time* $\text{poly}(N)$*, and for any fixed $k \in \mathbb{N}$, there are infinitely many integers n for which the following holds. Given any string $tt(f_n) \in \{0,1\}^N$ representing a function $f_n : \{0,1\}^n \to \{0,1\}$ computed by circuits in $\mathcal{C}[n^k]$, $\mathcal{A}$ outputs a circuit C on n inputs of size at most $s(n)$ that computes f_n with advantage $\delta(n)$.*

Proposition 8.5.3 (Lossy compression yields circuit lower bounds)**.**
Let $\mathcal{C}$ be a typical circuit class, and let $\delta(n) : \mathbb{N} \to (0, 1/2]$ be an arbitrary function. If there exists a $(\delta(n), o(2^n\delta^2/n))$-compression algorithm for $\mathcal{C}$, then $\mathsf{NEXP} \not\subseteq \mathcal{C}$.

Proof. Let $\mathcal{C}$ be a typical circuit class. Fix any function $\delta = \delta(n)$. Let $\mathcal{A}$ be an efficient $(\delta, o(2^n\delta^2/n))$-compression algorithm for $\mathcal{C}$. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that implicitly defines a property that is useful against $\mathcal{C}$. The proof then follows immediately from Proposition 8.4.2.

We define $\mathcal{B}$ as follows. Given any truth table $tt(f) \in \{0,1\}^N$ as input, apply $\mathcal{A}$ to $tt(f)$ to obtain the description of a circuit C over n inputs. If C is not a valid circuit, or it

has more than $\alpha \cdot 2^n\delta^2/n$ gates, accept. Otherwise, check if C computes f with advantage δ, and accepts $tt(f)$ if and only if this is *not* the case.

Let $\mathcal{P}$ be the property computed by $\mathcal{B}$. We need to check that $\mathcal{P}$ is a P-property that is useful against $\mathcal{C}$. First, observe that $\mathcal{B}$ runs in time $\text{poly}(N)$, since by assumption $\mathcal{A}$ is efficient, and $N = 2^n$. Also, B will always accept some family of hard functions, since it follows from Lemma 8.2.7 that for sufficiently large n there are functions that cannot be computed with advantage δ by circuits of size less than $\alpha \cdot 2^n\delta^2/n$. Finally, for any fixed k, it follows from the definition of lossy compression that there are infinitely many input sizes n on which $\mathcal{A}$ succeeds. For all such inputs sizes, algorithm $\mathcal{B}$ will correctly reject functions computed by circuits from $\mathcal{C}[n^k]$. □

This result is optimal for very small δ. More precisely, it follows from elementary Fourier analysis of Boolean functions that for every Boolean function f_n there is a parity function over some subset $S \subseteq [n]$ that computes f_n with advantage $\Omega(2^{-n/2})$. Further, it is possible to check all parity functions in deterministic time $\text{poly}(N)$.

Remark 9. *Similar techniques can be used to show that lossy compression of quasipolynomial size circuits leads to circuit lower bounds for* $\mathsf{NE} \cap \mathsf{i.o.coNE}$. *This can be obtained through an application of Proposition* 8.4.5.

8.5.2 Derandomization, SAT algorithms and circuit lower bounds

In this section we use Williams' framework to prove that derandomization yields circuit lower bounds. Recall that PIT is the language consisting of all arithmetic circuits that compute the zero polynomial over $\mathbb{Z}$, and PERM is the problem of computing the permanent of integer matrices.

Our proof uses the notion of useful algorithms introduced in Definition 8.3.9. The following consequence is immediate from Proposition 8.3.10.

Corollary 8.5.4. *Assume that* $\mathsf{NEXP} \subseteq \mathsf{SIZE}[\text{poly}]$. *Then there is* $c \in \mathbb{N}$ *such that there is no useful algorithm for* $\mathsf{Equiv\text{-}AND\text{-}SIZE}[n^c]$.

In addition, we will need the following auxiliary lemma.

Lemma 8.5.5 (Kabanets and Impagliazzo [110], Aaronson and van Melkebeek [1])**.**
There exists an efficient algorithm that takes as input an arithmetic circuit A_m and an integer m, and produces an arithmetic circuit C_m such that A_m computes the permanent of $m \times m$ matrices matrices over $\mathbb{Z}$ if and only if $C_m \in \mathsf{PIT}$.

We are now ready to give a short proof of the following result. Our argument follows the same high-level approach employed by [110] and [1].

Proposition 8.5.6 (Kabanets and Impagliazzo [110])**.**
If $\mathsf{PIT} \in \mathsf{NSUBEXP}$, then at least one of the following results hold:

(*i*) $\mathsf{NEXP} \not\subseteq \mathsf{SIZE}[\mathrm{poly}(n)]$*; or*

(*ii*) $\mathsf{PERM} \not\subseteq \mathsf{ASIZE}[\mathrm{poly}(n)]$.

Proof. In order to derive a contradiction, assume that:

- $\mathsf{PIT} \in \mathsf{NSUBEXP}$;
- $\mathsf{NEXP} \subseteq \mathsf{SIZE}[\mathrm{poly}(n)]$;
- $\mathsf{PERM} \subseteq \mathsf{ASIZE}[\mathrm{poly}(n)]$.

More precisely, $\mathsf{NEXP} \subseteq \mathsf{SIZE}[\mathrm{poly}(n)]$ implies that there exists a family of circuits $D = \{D_n\}_{n \in \mathbb{N}}$ of size n^d that solves $\mathsf{Equiv\text{-}AND\text{-}SIZE}[n^c]$. In addition, PERM over matrices of order m can be solved by a family of arithmetic circuits $A = \{A_m\}_{m \in \mathbb{N}}$ of size m^a (for some $a \in \mathbb{N}$). We prove that these assumptions contradict Corollary 8.5.4. We construct a useful algorithm $\mathcal{A}$ for $\mathsf{Equiv\text{-}AND\text{-}SIZE}[n^c]$ as follows.

Algorithm $\mathcal{A}$:
Input: Circuits C_1, C_2 of size n^c.

- First, $\mathcal{A}$ guesses a circuit D_n of size n^d.
- $\mathcal{A}$ prepares a query to the polynomial time hierarchy[13] to check if D_n solves $\mathsf{Equiv\text{-}AND\text{-}SIZE}[n^c]$.

[13] Observe that D_n does not solve the equivalence problem if and only if ($\exists C_1, C_2\ \exists x$ such that $C_1(x) \neq C_2(x)$ and $D_n(C_1, C_2) = 1$) or ($\exists C_1, C_2$ such that $\forall x (C_1(x) = C_2(x))$ and $D_n(C_1, C_2) = 0$).

- It uses Toda's theorem [185] together with the completeness of the permanent problem [191] to reduce this query to a call to PERM over matrices of dimension m, where $m = \text{poly}(n^d)$.
- Next, $\mathcal{A}$ guesses an arithmetic circuit A_m of size m^a.
- It then applies Lemma 8.5.5 to obtain a circuit C_m such that A_m computes the permanent of $m \times m$ matrices matrices over $\mathbb{Z}$ if and only if $C_m \in \mathsf{PIT}$.
- Now $\mathcal{A}$ uses nondeterminism and the assumption that $\mathsf{PIT} \in \mathsf{NSUBEXP}$ to check if $C_m \in \mathsf{PIT}$. It aborts otherwise.
- It uses A_m to answer the initial query, and aborts if D_n does not solve Equiv-AND-SIZE$[n^c]$.
- Finally, $\mathcal{A}$ uses D_n to solve Equiv-AND-SIZE$[n^c]$ on inputs C_1 and C_2.

Clearly, $\mathcal{A}$ runs in nondeterministic subexponential time. In addition, it is easy to see that it is a useful algorithm for Equiv-AND-SIZE$[n^c]$, which completes the proof of Proposition 8.1.6. □

Most importantly, this proof shows that any improvement over Corollary 8.5.4 implies a corresponding improvement over Proposition 8.5.6. In addition, it is not hard to see that Conjecture 8.4.6 immediately implies the extension of Proposition 8.5.6 obtained by Aaronson and van Melkebeek [1].[14]

8.5.3 Useful properties and learning algorithms

The existence of learning algorithms in many different models yields circuit lower bounds, as shown by Fortnow and Klivans [70]. In this section we discuss two frameworks for learning: deterministic exact learning from membership and equivalence queries (Angluin [17]), and randomized PAC learning (Valiant [193]). Recall that we have discussed

[14] Here is a sketch of the argument. Assume that $\mathsf{NE} \cap \mathsf{coNE} \subseteq \mathsf{P/poly}$. Then by Conjecture 8.4.6 there is no P-property useful against P/poly. However, it is possible to show that useful algorithms for satisfiability lead to useful properties. Altogether, these assumptions imply the desired strengthening of Corollary 8.5.4.

several learning models in Chapter 5. For convenience of the reader, we briefly review the definitions that will be relevant for the results discussed here.

Exact learning algorithms. Let $\mathcal{C}$ be a typical circuit class. In this model, a *deterministic* algorithm is given access to oracles MQ^f and EQ^f for some function $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{C}$. There oracles are defined as follows.

MQ^f: Given $x \in \{0,1\}^n$, returns $f(x)$.

EQ^f: Given a hypothesis $h : \{0,1\}^n \to \{0,1\}$ represented as a circuit, returns 1 if $h \equiv f$. Otherwise, returns an arbitrary input $x \in \{0,1\}^n$ such that $f(x) \neq h(x)$.

For a size function $s : \mathbb{N} \to \mathbb{N}$, we say that a learning algorithm $\mathcal{A}$ *exact learns* $\mathcal{C}[s(n)]$ in time $t(n)$ if for every $f \in \mathcal{C}$, when given access to oracles MQ^f and EQ^f, $\mathcal{A}$ runs in time at most $t(n)$, and outputs the description of a circuit C computing f. In particular, every equivalence query is invoked on a circuit of size at most $t(n)$, and the final hypothesis C is a circuit of size at most $t(n)$.

Recall that one of the main results from Fortnow and Klivans [70] states that exact learning a circuit class leads to circuit lower bounds against E^{NP} (Proposition 8.1.7). The original proof used by them is a clever combination of many results from complexity theory. Here we observe that it is relatively easy to prove results of this form using the machinery of useful properties. To simplify the argument even more, we can view learning as compression, which yields a quick proof of the following result.

Proposition 8.5.7 ("Learning yields circuit lower bounds")**.**
Let $\mathcal{C}$ be a circuit class. Suppose there exists an exact learning algorithm for $\mathcal{C}$[poly] *that runs in subexponential time. Then* $\mathsf{NEXP} \not\subseteq \mathcal{C}$[poly].

Proof. Let $\mathcal{A}$ be an exact learning algorithm for $\mathcal{C}$. It is easy to see that given any truth-table $tt(h) \in \{0,1\}^N$ from $\mathcal{C}$[poly], we can simulate $\mathcal{A}$ on input h in time $2^{O(n)}$. In other words, it is possible to provide correct answers to the membership and equivalence queries asked during $\mathcal{A}$'s computation. By assumption, the learning algorithm outputs a circuit of subexponential size that computes h. This is therefore a valid compression algorithm for

$\mathcal{C}$[poly], and Proposition 8.5.7 follows immediately from Proposition 8.5.3 with $\delta = 1$. □

In addition to its simplicity, this proof offers other advantages. The framework of useful properties is more flexible with respect to changes in the learning model. For instance, one could consider deterministic learning algorithms using equivalence queries over subsets $S \subseteq \{0,1\}^n$ encoded by subexponential size circuits, and only require that the learning algorithm outputs a hypothesis that is ε-close to the unknown concept. Again, Proposition 8.5.3 easily implies circuit lower bounds.

Next we turn our attention to randomized learning algorithms, a class of algorithms for which theorems of the form "learning implies circuit lower bounds" are still a bit weaker than their deterministic counterpart.

Randomized PAC learning algorithms. In the PAC learning framework, there is an unknown function $f \in \mathcal{C}$ that the learning algorithm is supposed to learn (after obtaining limited information about f). Here we concentrate on the stronger model in which the learner can ask membership queries, and only needs to learn under the uniform distribution[15]. In other words, the learner can query the value $f(x)$ on any input x, and should be able to obtain, with high probability, a good approximation h for f. In general, for any function $f : \{0,1\}^n \to \{0,1\}$ in $\mathcal{C}[s(n)]$, given parameters n, ε (accuracy), δ (confidence), and an upper bound $s(n)$ on the size of the circuit computing f, the learning algorithm should output with probability at least $1-\delta$ a hypothesis h such that $\Pr_x[f(x) \neq h(x)] \leq \varepsilon$ (i.e., h is ε-close to f), where the probability is taken over all strings x of size n under the uniform distribution. We measure the running time $t_{\mathcal{A}}(n, 1/\delta, 1/\varepsilon, s(n))$ of a learning algorithm $\mathcal{A}$ as a function of these parameters. As opposed to what is usually called proper learning, the learning algorithm is allowed to output the description of any circuit of size at most $t_{\mathcal{A}}(.)$ as its final hypothesis. For simplicity, we say that an algorithm $\mathcal{A}$ PAC learns $\mathcal{C}$ if it learns any function from $\mathcal{C}$ to accuracy $1/4$ with probability at least $1 - 1/n$.

It is known that the existence of a polynomial time PAC learning algorithm for $\mathcal{C}$[poly]

[15] In other words, a transference theorem for this learning model is a stronger result. In addition, it is easy to see that the results discussed here hold under even more powerful learning models.

implies that $\mathsf{BPEXP} \not\subseteq \mathcal{C}[\text{poly}]$ (Fortnow and Klivans [70]). However, the same proof provides much weaker results for subexponential time learning, and it is an interesting open problem to show that the existence of subexponential time PAC learning algorithms lead to similar circuit lower bounds. The next proposition shows that this problem is related to the power of randomness in the context of useful properties. First, we extend the definition of useful properties to promise properties.

Definition 8.5.8 ("Promise properties useful against $\mathcal{C}$")**.**
A promise property *of Boolean functions $\mathcal{P} = (\mathcal{P}_{\mathsf{yes}}, \mathcal{P}_{\mathsf{no}})$ consists of two nonempty disjoint subsets of the set of all Boolean functions. For a typical circuit class $\mathcal{C}$, $\mathcal{P}$ is said to be* useful *against $\mathcal{C}$ if, for all k, there are infinitely many positive integers n such that*

- $\mathcal{P}_{\mathsf{yes}}(f) = 1$ *for at least one function $f : \{0,1\}^n \to \{0,1\}$, and*
- $\mathcal{P}_{\mathsf{no}}(g) = 1$ *for all $g : \{0,1\}^n \to \{0,1\}$ that admits circuits from $\mathcal{C}[n^k]$.*

We say that a promise property $\mathcal{P}$ is a Γ-property *if its corresponding promise problem $L_{\mathcal{P}}$ is in* promise-Γ.

Proposition 8.5.9 ("Useful properties from randomized learning")**.**
Let $\mathcal{C}$ be a typical circuit class. Suppose there exists a randomized algorithm $\mathcal{A}$ that PAC *learns $\mathcal{C}$*[poly] *in time $2^{n^{o(1)}}$. Then there exists a* (promise-coRP)*-property that is useful against $\mathcal{C}$.*

Proof. We use a subexponential time randomized learning algorithm $\mathcal{A}$ for $\mathcal{C}$ to define a (promise) coRP-property $\mathcal{P}$ that is useful against $\mathcal{C}$. Consider the following randomized algorithm $\mathcal{B}$. Given the truth-table $tt(f_n) \in \{0,1\}^N$ of an arbitrary function $f_n : \{0,1\}^n \to \{0,1\}$, it simulates the computation of $\mathcal{A}$ over f_n, until $\mathcal{A}$ outputs a circuit C of size $2^{n^{o(1)}}$ as its final hypothesis. Algorithm $\mathcal{B}$ accepts f_n if and only if C is *not* 1/10-close to f_n.

It follows from Lemma 8.2.7 that for any large enough n there is a function h_n that cannot be 1/10-approximated by circuits of subexponential size (for definiteness, fix some constructive size bound). In other words, for any large n, there exists at least one function h_n not in $\mathcal{C}$[poly] that is accepted with probability one. In addition, since $\mathcal{A}$ is a PAC learning algorithm for $\mathcal{C}$, every function in $\mathcal{C}$ is rejected with high probability. Clearly,

$\mathcal{B}$ computes a promise coRP-property that is useful against $\mathcal{C}$: $\mathcal{P}_{\mathsf{yes}}$ consists of Boolean functions that cannot be approximated by circuits of subexponential size, and $\mathcal{P}_{\mathsf{no}} = \mathcal{C}$. □

This result gives another example of the relevance of useful properties in the context of results of the form "algorithms yield circuit lower bounds".

8.6 Open problems and further research directions

We mention here three directions related to the results discussed in this chapter that we find particularly interesting.

Strengthening the ACC lower bound. Williams proved that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}$. It follows easily from Lemma 8.2.2 that either $\mathsf{P} \not\subseteq \mathsf{ACC}$ or $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. Give an unconditional proof that one of these circuit lower bounds hold.

Stronger lower bound from satisfiability algorithms. Can we prove that the existence of non-trivial (deterministic) satisfiability algorithms for a circuit class $\mathcal{C}$ leads to lower bounds for complexity classes contained in $\mathsf{NEXP} \cap \mathsf{coNEXP}$?

Lossy compression of ACC and TC^0_2. Design efficient lossy compression schemes for circuit classes such as ACC or TC^0_2. To the best of our knowledge, these results do not violate any widely believed cryptographic assumption.

8.7 Auxiliary results

In this section we describe the proof of Proposition 8.4.5, which we state again for convenience.

Proposition. *Let $\mathcal{C}$ be a typical circuit class. If for every $c \in \mathbb{N}$ there exists a P-property that is useful against $\mathcal{C}[n^{\log^c n}]$, then $\mathsf{NE} \cap \mathsf{i.o.coNE} \not\subseteq \mathcal{C}[n^{\log n}]$.*

This result is implicit in the work of Williams [201]. Its proof consists of an interesting combination of nondeterminism, a collapse theorem, a hardness vs. randomness result, and simple diagonalization. We will need the following auxiliary results.

Lemma 8.7.1. *Let $\mathcal{C}$ be a typical circuit class, and assume that $\mathsf{P} \subseteq \mathcal{C}[n^{\log n}]$. Then for every $d \in \mathbb{N}$, any function $f : \{0,1\}^n \to \{0,1\}$ computed by circuits of size $n^{\log^d n}$ is computed by circuits from $\mathcal{C}[n^{\log^{O(d)} n}]$.*

Proof. The result follows from a parameterized version of Lemma 8.2.2, and the proof is similar. □

Lemma 8.7.2 (Miltersen, Vinodchandran and Watanabe [137])**.**
Let $g(n) > 2^n$ and $s(n) \geq n$ be functions that are both increasing and time-constructible. There exists a constant $d \in \mathbb{N}$ for which the following holds. If $\mathsf{E} \subseteq \mathsf{SIZE}(s(n))$ then $\mathsf{DTIME}[g(n)] \subseteq \mathsf{MATIME}[s(d \log g(n))^d]$.

For a function $h_\ell : \{0,1\}^\ell \to \{0,1\}$, let $\mathsf{CC}(h)$ be the size (number of gates) of the smallest circuit computing h.

Proposition 8.7.3 (Umans [188])**.**
There is a constant $k \in \mathbb{N}$ and a function $G : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ for which the following holds. For every $s \in \mathbb{N}$ and Boolean function $h_\ell : \{0,1\}^\ell \to \{0,1\}$ satisfying $\mathsf{CC}(h_\ell) \geq s^k$, and for all circuits C of size at most s over s inputs,

$$\left| \Pr_{z \in \{0,1\}^{k \cdot \ell}} [C(G(tt(h_\ell), z)) = 1] - \Pr_{z \in \{0,1\}^s} [C(z) = 1] \right| < \frac{1}{s}.$$

In addition, G can be computed in $\mathsf{poly}(2^\ell)$ time.

The next lemma shows that useful properties together with the lack of circuit lower bounds for P allow us to obtain a nontrivial derandomization of Merlin-Arthur games.

Lemma 8.7.4. *Let $\mathcal{C}$ be a typical circuit class, and suppose that for every $c \in \mathbb{N}$ there exists a P-property that is useful against $\mathcal{C}[n^{\log^c n}]$. In addition, assume that $\mathsf{P} \subseteq \mathcal{C}[n^{\log n}]$. Then there is an infinite subset $S \subseteq \mathbb{N}$ such that for any $L \in \mathsf{MATIME}[n^{O(\log^3 n)}]$, there exists a language $L' \in \mathsf{NE}$ such that for every $n \in S$, we have $L_n = L'_n$. In addition, for all $n \notin S$, we have $L'_n = \emptyset$.*

Proof. First, observe that Lemma 8.7.1 implies that for every $c \in \mathbb{N}$ there exists a property $\mathcal{P}_c$ that is useful against $\mathsf{SIZE}[n^{\log^c n}]$. Let $\mathcal{A}^c$ be an efficient algorithm computing $\mathcal{P}_c$ (we

set the value of c later). Let $L \in \mathsf{MATIME}[n^{O(\log^3 n)}]$. There exists a MA-verifier V for L running in time $s = n^{O(\log^3 n)}$ such that

$$x \in L \implies \exists y \in \{0,1\}^s \Pr_{w \in \{0,1\}^s}[V(x,y,w) = 1] \geq \frac{2}{3}$$
$$x \notin L \implies \forall y \in \{0,1\}^s \Pr_{w \in \{0,1\}^s}[V(x,y,w) = 1] \leq \frac{1}{3}$$

Our nondeterministic algorithm $\mathcal{N}$ for L proceeds as follows. On input $x \in \{0,1\}^n$, it first guesses a string $y \in \{0,1\}^s$, then constructs a circuit $C_{x,y}$ from $\mathsf{SIZE}[s]$ such that for all $w \in \{0,1\}^s$ we have $C_{x,y}(w) = V(x,y,w)$. Then $\mathcal{N}$ guesses truth-tables $tt(h_m) \in \{0,1\}^M$ for every $m \in [2^{(\log n)^{5/(c+1)}}, 2^{(\log(n+1))^{5/(c+1)}})$, where $M = 2^m$ as usual. If $\mathcal{A}^c$ rejects all such functions, then $\mathcal{N}$ rejects x. Otherwise, let h_ℓ be the first function for which $\mathcal{A}^c(h_\ell) = 1$. Since $\mathcal{A}^c$ computes a useful property, $s = n^{O(\log^3 n)}$, and $\ell \geq 2^{(\log n)^{5/(c+1)}}$, for any $c \in \mathbb{N}$ we have:

$$\mathsf{CC}(h_\ell) \geq \ell^{\log^c \ell} \geq n^{\log^4 n} \gg s^k,$$

for any $k \in \mathbb{N}$ and sufficiently large n. Finally, $\mathcal{N}$ runs the algorithm granted by Proposition 8.7.3 on $C_{x,y}$ using h_ℓ, and accepts its input x if and only if

$$\Pr_{z \in \{0,1\}^{k \cdot \ell}}[C_{x,y}(G(tt(h_\ell), z)) = 1] \geq \frac{1}{2}. \tag{8.3}$$

Observe that there exists an infinite set $S \subseteq \mathbb{N}$ such that for each $n \in S$ and for every $x \in \{0,1\}^n$, $\mathcal{N}$ is able to find a function h_ℓ for which $\mathsf{CC}(h_\ell) \geq s^k$, where k is the constant in the statement of Proposition 8.7.3. Put another way, $\mathcal{N}$ is correct on input sizes in S, and by construction $\mathcal{N}$ rejects every other input whose input size is not in S. Also, S depends only on $\mathcal{P}_c$.

The (nondeterministic) running time of $\mathcal{N}$ is dominated by the computation of the probability in (8.3), and the time required to verify using $\mathcal{A}^c$ whether some hard function has been guessed. Finally, set $c = 5$, and observe that for this value of c we have $\ell \ll n$. It follows therefore that $\mathcal{N}$ runs in time at most 2^n. This completes the proof that there exists $L' \in \mathsf{NE}$ such that for every $n \in S$, $L'_n = L_n$, and for all $n \notin S$, we have $L'_n = \emptyset$. □

We are now ready to give the proof of Proposition 8.4.5.

Proof of Proposition 8.4.5. Assume that $\mathsf{NE} \cap \mathsf{i.o.coNE} \subseteq \mathcal{C}[n^{\log n}]$. In particular, $\mathsf{E} \subseteq \mathsf{SIZE}[n^{\log n}]$. Let $g(n) = 2^{n^2 \log n}$ and $s(n) = n^{\log n}$. Using Lemma 8.7.2, we get $\mathsf{DTIME}[2^{n^2 \log n}] \subseteq \mathsf{MATIME}[n^{O(\log^3 n)}]$. Clearly, our assumptions also imply that $\mathsf{P} \subseteq \mathcal{C}[n^{\log n}]$.

Let $L \in \mathsf{DTIME}[2^{n^2 \log n}]$. It follows from Lemma 8.7.4 that there exists an infinite set $S \subseteq \mathbb{N}$ and a language $L' \in \mathsf{NE}$ such that $L_n = L'_n$ for every $n \in S$. Consider $\overline{L} \in \mathsf{DTIME}[2^{n^2 \log n}]$, the complement of L. Then, again, there exists a language $L'' \in \mathsf{NE}$ such that for every $n \in S$, $\overline{L}_n = L''_n$. Clearly, $\overline{L''} \in \mathsf{coNE}$, and for every $n \in S$ we have $\overline{L''}_n = L_n = L'_n$. In other words, $L' \in \mathsf{NE} \cap \mathsf{i.o.coNE}$. Overall, we get

$$\mathsf{DTIME}[2^{n^2 \log n}] \subseteq \mathsf{i.o.}(\mathsf{NE} \cap \mathsf{i.o.coNE}) \subseteq \mathsf{i.o.}\mathcal{C}[n^{\log n}],$$

where the last inclusion uses our initial assumption.

However, using a simple diagonalization argument, we can define a language $L^* \in \mathsf{DTIME}[2^{n^2 \log n}]$ such that for all $n \geq n_0$, L^*_n is not computed by circuits from $\mathcal{C}[n^{\log n}]$. This contradiction completes the proof of Proposition 8.4.5. □

Chapter 9

Concluding remarks

The results presented in this work encompass the power and limitations of bounded-depth circuits and monotone circuits, the role of negations in learning theory and cryptography, and some connections between algorithms and circuit lower bounds. Our proofs combine and extend many techniques employed in theoretical computer science in the past, including recent approaches to circuit lower bounds. We discussed a few concrete open problems and research directions in each appropriate chapter.

The main challenge lies in understanding more general classes of Boolean circuits. As far as unconditional lower bounds are concerned, their computational power remains mysterious, and it is unclear which mathematical techniques will turn out to be useful in the investigation of these problems. We believe that the interplay between unconditional lower bounds, conditional results, and algorithm design will continue to shed light into this research area.

Finally, one should not be discouraged by the difficulty of proving unconditional lower bounds for general algorithms and circuit classes. Computational complexity theory is a relatively young discipline, and we share the hope and excitement that a satisfactory answer to these problems will eventually be discovered, similarly to many other seemingly unapproachable problems from different domains of Mathematics.

Bibliography

[1] Scott Aaronson and Dieter van Melkebeek. On circuit lower bounds from derandomization. *Theory of Computing*, 7(1):177–184, 2011. 186, 191, 197, 215, 216

[2] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *Transactions on Computation Theory*, 1(1), 2009. 50, 189

[3] Scott Aaronson, Baris Aydinlioglu, Harry Buhrman, John M. Hitchcock, and Dieter van Melkebeek. A note on exponential circuit lower bounds from derandomizing Arthur-Merlin games. *Electronic Colloquium on Computational Complexity* (ECCC), 17:174, 2010. 186

[4] Leonard M. Adleman. Two theorems on random polynomial time. In *Symposium on Foundations of Computer Science* (FOCS), pages 75–83, 1978. 171

[5] Alok Aggarwal, Maria M. Klawe, David Lichtenstein, Nathan Linial, and Avi Wigderson. A lower bound on the area of permutation layouts. *Algorithmica*, 6(2):241–255, 1991. 46

[6] Miklós Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Symposium on Theory of Computing* (STOC), pages 471–474, 1984. 76

[7] Miklós Ajtai and Yuri Gurevich. Monotone versus positive. *J. ACM*, 34(4):1004–1015, 1987. 26

[8] Miklós Ajtai. $\sum_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983. 50

[9] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate

weak pseudorandom functions in $\mathrm{AC}^0 \circ \mathrm{MOD}_2$. In *Innovations in Theoretical Computer Science* (ITCS), pages 251–260, 2014. 147

[10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3), 2010. 51, 189

[11] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 1–15, 2001. 194

[12] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987. 50

[13] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley-Interscience, 2008. 11

[14] Kazuyuki Amano and Akira Maruoka. On learning monotone Boolean functions under the uniform distribution. In *International Conference on Algorithmic Learning Theory* (ALT), pages 57–68, 2002. 98

[15] Kazuyuki Amano and Akira Maruoka. A superpolynomial lower bound for a circuit computing the clique function with at most $1/6 \log \log n$ negation gates. *SIAM J. Comput.*, 35(1):201–216, 2005. 100, 120

[16] Alexander E. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Math. Dokl*, 31(3):530–534, 1985. 50

[17] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988. 98, 160, 163, 216

[18] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. 120, 147

[19] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. 58, 125, 198

[20] Sanjeev Arora, Russell Impagliazzo, and Umesh Vazirani. Relativizing versus nonrelativizing techniques: the role of local checkability. *Manuscript*, 1992. 189

[21] Baris Aydinlioglu, Dan Gutfreund, John M. Hitchcock, and Akinori Kawachi. Derandomizing Arthur-Merlin games and approximate counting implies exponential-size lower bounds. *Computational Complexity*, 20(2):329–366, 2011. 186

[22] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. 172

[23] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993. 153

[24] Theodore P. Baker, John Gill, and Robert Solovay. Relativizatons of the $\mathsf{P} =? \mathsf{NP}$ question. *SIAM J. Comput.*, 4(4):431–442, 1975. 50, 189

[25] Robert Beals, Tetsuro Nishino, and Keisuke Tanaka. More on the complexity of negation-limited circuits. In *Symposium on Theory of Computing* (STOC), pages 585–595, 1995. 120

[26] Robert Beals, Tetsuro Nishino, and Keisuke Tanaka. On the complexity of negation-limited Boolean networks. *SIAM J. Comput.*, 27(5):1334–1347, 1998. 120

[27] Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. Approximating AC^0 by small height decision trees and a deterministic algorithm for $\#\mathrm{AC}^0$. In *Conference on Computational Complexity* (CCC), pages 117–125, 2012. 187

[28] Eric Blais and Li-Yang Tan. Approximating Boolean functions with depth-2 circuits. *Electronic Colloquium on Computational Complexity* (ECCC), 20:51, 2013. 111

[29] Eric Blais, Clément L. Canonne, Igor C. Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. *Electronic Colloquium on Computational Complexity* (ECCC), 21:144, 2014. 120, 124, 129, 138, 140, 141, 148, 150

[30] Avrim Blum, Merrick L. Furst, Jeffrey Jackson, Michael J. Kearns, Yishai Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Symposium on Theory of Computing* (STOC), pages 253–262, 1994. 9, 157, 177

[31] Avrim Blum, Carl Burch, and John Langford. On learning monotone Boolean functions. In *Symposium on Foundations of Computer Science* (FOCS), pages 408–415, 1998. 98, 101, 120, 121, 138

[32] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, 1987. 156, 171

[33] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. 53

[34] Hans L. Bodlaender. NC-algorithms for graphs with small treewidth. In *International Workshop on Graph-Theoretic Concepts in Computer Science* (WG), pages 1–10, 1988. 23, 44

[35] Hans L. Bodlaender. Treewidth: Characterizations, applications, and computations. In *Workshop on Graph-Theoretic Concepts in Computer Science* (WG), pages 1–14, 2006. 16

[36] Andrej Bogdanov and Siyao Guo. Sparse extractor families for all the entropy. In *Innovations in Theoretical Computer Science* (ITCS), pages 553–560, 2013. 121, 123, 141

[37] Lucas Bordeaux, Youssef Hamadi, and Pushmeet Kohli. *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 2014. 23

[38] Allan Borodin. Horner's rule is uniquely optimal. In *International Symposium on the Theory of Machines and Computations*, pages 45–57, 1971. 91

[39] Nader H. Bshouty and Vitaly Feldman. On using extended statistical queries to avoid membership queries. *The Journal of Machine Learning Research*, 2:359–395, 2002. 162

[40] Nader H. Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996. 98, 101, 105, 110, 120

[41] Nader H. Bshouty, Richard Cleve, Ricard Gavaldà, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *J. Comput. Syst. Sci.*, 52(3):421–433, 1996. 157

[42] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: A survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002. 21

[43] Harry Buhrman and John M. Hitchcock. NP-hard sets are exponentially dense unless $\mathsf{coNP} \subseteq \mathsf{NP/poly}$. In *Conference on Computational Complexity* (CCC), pages 1–7, 2008. 53

[44] Joshua Buresh-Oppenheim, Valentine Kabanets, and Rahul Santhanam. Uniform hardness amplification in NP via monotone codes. *Electronic Colloquium on Computational Complexity* (ECCC), 13(154), 2006. 121, 123, 138

[45] Arkadev Chattopadhyay and Rahul Santhanam. Lower bounds on interactive compressibility by constant-depth circuits. In *Symposium on Foundations of Computer Science* (FOCS), pages 619–628, 2012. 7, 51, 53, 54, 57, 65, 66, 70, 81, 83, 91

[46] Eshan Chattopadhyay, Adam Klivans, and Pravesh Kothari. An explicit VC-theorem for low-degree polynomials. In *Workshop on Randomization and Computation* (RANDOM), pages 495–504, 2012. 158, 173, 174

[47] Bernard Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, 2000. 173

[48] Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. In *Symposium on Theory of Computing* (STOC), pages 60–69, 2014. 25, 46

[49] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Electronic Colloquium on Computational Complexity* (ECCC), 20:57, 2013. 186, 187, 192, 193, 195, 213

[50] Timothy Y. Chow. Almost-natural proofs. *J. Comput. Syst. Sci.*, 77(4):728–737, 2011. 190

[51] Amin Coja-Oghlan. The asymptotic k-SAT threshold. In *Symposium on Theory of Computing* (STOC), pages 804–813, 2014. 19

[52] Stephen A. Cook. A hierarchy for nondeterministic time complexity. *J. Comput. Syst. Sci.*, 7(4):343–353, 1973. 204

[53] Michal Cutler and Yossi Shiloach. Permutation layout. *Networks*, 8(3):253–278, 1978. 46

[54] Dana Dachman-Soled, Homin K. Lee, Tal Malkin, Rocco A. Servedio, Andrew Wan, and Hoeteck Wee. Optimal cryptographic hardness of learning monotone functions. *Theory of Computing*, 5(1):257–282, 2009. 120

[55] Evgeny Dantsin and Edward A. Hirsch. Worst-case upper bounds. In *Handbook of Satisfiability*, pages 403–424. 2009. 187

[56] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23, 2014. 53

[57] Holger Dell. A simple proof that AND-compression of NP-complete problems is hard. *Electronic Colloquium on Computational Complexity* (ECCC), 14:75, 2014. 53

[58] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. 15

[59] Andrew Drucker. New limits to classical and quantum instance compression. In *Symposium on Foundations of Computer Science* (FOCS), pages 609–618, 2012. 53

[60] Bella Dubrov and Yuval Ishai. On the randomness complexity of efficient sampling. In *Symposium on Theory of Computing* (STOC), pages 711–720, 2006. 52

[61] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *Inter-*

national Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), pages 135–156, 2010. 53

[62] Uriel Feige. Relations between average case complexity and approximation complexity. In *Symposium on Theory of Computing* (STOC), pages 534–543, 2002. 19

[63] Vitaly Feldman, Homin K. Lee, and Rocco Servedio. Lower bounds and hardness amplification for learning shallow monotone formulas. *Journal of Machine Learning Research*, 19:273–292, 2011. 109

[64] Vitaly Feldman. Evolvability from learning algorithms. In *Symposium on Theory of Computing* (STOC), pages 619–628, 2008. 157, 161, 174

[65] William Feller. Generalization of a probability limit theorem of Cramér. *Transactions of the American Mathematical Society*, 54(3):361–372, 1943. 93

[66] Michael J. Fischer. The complexity of negation-limited networks - A brief survey. In *International Colloquium on Automata, Languages and Programming* (ICALP), pages 71–82, 1975. 121

[67] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. 15, 38, 40, 44

[68] Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans-Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 171–182, 2001. 82

[69] Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. Syst. Sci.*, 65(4):612–625, 2002. 82

[70] Lance Fortnow and Adam R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009. 9, 152, 154, 156, 167, 168, 169, 181, 186, 191, 192, 196, 216, 217, 219

[71] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. 53

[72] Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):835–865, 2005. 203

[73] Cees M. Fortuin, Pieter W. Kasteleyn, and Jean Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22(2):89–103, 1971. 130

[74] Alan M. Frieze. Edge-disjoint paths in expander graphs. *SIAM J. Comput.*, 30(6):1790–1801, 2000. 32

[75] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. 50, 52, 189

[76] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Inf. Process. Lett.*, 43(4):169–174, 1992. 169

[77] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *Symposium on Foundations of Computer Science* (FOCS), pages 107–109, 2011. 152

[78] Mikael Goldmann and Alexander Russell. Spectral bounds on general hard-core predicates. In *Symposium on Theoretical Aspects of Computer Science* (STACS), pages 614–625, 2000. 121, 123, 140

[79] Oded Goldreich and Rani Izsak. Monotone circuits: One-way functions versus pseudorandom generators. *Theory of Computing*, 8(1):231–238, 2012. 8, 120, 121, 122, 131, 133

[80] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Symposium on Theory of Computing* (STOC), pages 25–32, 1989. 119

[81] Oded Goldreich and David Zuckerman. Another proof that BPP $\subseteq$ PH (and more). In *Studies in Complexity and Cryptography*, pages 40–53. 2011. 172

[82] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. 119

[83] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2007. 125

[84] Oded Goldreich. *Computational Complexity - A Conceptual Perspective*. Cambridge University Press, 2008. 198

[85] Oded Goldreich. In a world of P=BPP. In *Studies in Complexity and Cryptography*, pages 191–232. 2011. 191

[86] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity (Full Version). *CoRR*, abs/1311.2355, 2013. 31

[87] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity (Conference Version). In *Symposium on Theory of Computing* (STOC), pages 847–856, 2014. 6, 15, 19, 20, 21, 22, 31

[88] Parikshit Gopalan and Rocco A. Servedio. Learning and lower bounds for AC^0 with threshold gates. In *Workshop on Randomization and Computation* (RANDOM), pages 588–601, 2010. 86

[89] Martin Grohe, Thomas Schwentick, and Luc Segoufin. When is the evaluation of conjunctive queries tractable? In *Symposium on Theory of Computing* (STOC), pages 657–666, 2001. 16

[90] Martin Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1), 2007. 6, 16, 24

[91] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *J. Comput. Syst. Sci.*, 46(2):129–154, 1993. 57

[92] Ryan C. Harkins and John M. Hitchcock. Exact learning algorithms, betting games, and circuit lower bounds. In *International Colloquium on Automata, Languages and Programming* (ICALP), pages 416–423, 2011. 9, 153, 154, 167, 186, 192

[93] Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010. 53

[94] Juris Hartmanis and Richard Stearns. Classification of computations by time and memory requirements. In *Proceedings of the* IFIP *Congress*, volume 65, pages 31–35, 1965. 2

[95] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Symposium on Theory of Computing* (STOC), pages 6–20, 1986. 5, 50, 52, 86

[96] Johan Håstad. On the correlation of parity and small-depth circuits. *SIAM J. Comput.*, 43(5):1699–1708, 2014. 5

[97] Thomas Hofmeister. The power of negative thinking in constructing threshold circuits for addition. In *Structure in Complexity Theory Conference* (CCC), pages 20–26, 1992. 26, 120, 147

[98] Johan Håstad. *Computational Limitations for Small Depth Circuits*. PhD thesis, MIT, 1986. 189

[99] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: amplifying communication complexity hardness to time-space trade-offs in proof complexity. In *Symposium on Theory of Computing* (STOC), pages 233–248, 2012. 21, 22

[100] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. 25

[101] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing* (STOC), pages 220–229, 1997. 167

[102] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. 191

[103] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. 19

[104] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002. 191, 201

[105] Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. An axiomatic approach to algebrization. In *Symposium on Theory of Computing* (STOC), pages 695–704, 2009. 189

[106] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Symposium on Discrete Algorithms* (SODA), pages 961–972, 2012. 187

[107] Russell Impagliazzo, Ramamohan Paturi, and Stefan Schneider. A satisfiability algorithm for sparse depth-2 threshold circuits. *CoRR*, abs/1212.4548, 2012. 187, 207

[108] Hamidreza Jahanjou, Eric Miles, and Emanuele Viola. Local reductions. *Electronic Colloquium on Computational Complexity* (ECCC), 20:20, 2013. 194

[109] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*. Springer, 2012. 1, 4, 5, 14, 17, 121, 125

[110] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. 186, 191, 215

[111] Valentine Kabanets. Derandomization: A brief overview. *Bulletin of the EATCS*, 76:88–103, 2002. 191

[112] Valentine Kabanets. Private communication, 2013. 155, 192

[113] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions. In *Symposium on Foundations of Computer Science* (FOCS), pages 68–80, 1988. 131

[114] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990. 12, 19

[115] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Symposium on Theory of Computing* (STOC), pages 302–309, 1980. 171, 185, 187

[116] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994. 98, 152

[117] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994. 1, 4, 184

[118] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. 161, 177

[119] V. M. Khrapchenko. A method of determining lower bounds for the complexity of π-schemes. *Math. Notes Acad. of Sci.* (USSR), 10(1):474–479, 1971. 52

[120] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Computational Complexity*, 21(1):3–61, 2012. 186, 191

[121] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12, 2009. 152, 156

[122] Adam Klivans, Pravesh Kothari, and Igor C. Oliveira. Constructing hard functions using learning algorithms. In *Conference on Computational Complexity* (CCC), pages 86–97, 2013. 186, 192

[123] Swastik Kopparty and Srikanth Srinivasan. Certifying polynomials for $\mathrm{AC}^0(\oplus)$ circuits, with applications. In *Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 36–47, 2012. 55, 66, 95, 96

[124] Sajin Koroth and Jayalal Sarma. Depth lower bounds against circuits with sparse orientation. In *Conference on Computing and Combinatorics* (COCOON), pages 596–607, 2014. 124, 126, 145, 150

[125] Aleksej D. Korshunov. Monotone Boolean functions. *Russian Mathematical Surveys*, 58(5):929–1001, 2003. 14, 98

[126] Jan Krajícek. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge University Press, 1995. 1, 4

[127] Matthias Krause and Stefan Lucks. Pseudorandom functions in TC^0 and cryptographic limitations to proving lower bounds. *Computational Complexity*, 10(4):297–313, 2001. 190

[128] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997. 1, 4, 12, 20, 52, 58, 125

[129] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993. 105, 187

[130] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *Machine Learning*, pages 285–318, 1988. 163

[131] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977. 140

[132] A. A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, 1958. 8, 99, 121, 122, 128, 129, 150

[133] Dániel Marx. Can you beat treewidth? *Theory of Computing*, 6(1):85–112, 2010. 16, 26

[134] Dániel Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):42, 2013. 16

[135] Jiří Matoušek and Jan Vondrák. *Lecture notes on the probabilistic method*, 2008. 93

[136] Jiří Matoušek. *Geometric Discrepancy: An Illustrated Guide* (*Algorithms and Combinatorics*). Springer, 1999. 173

[137] Peter Bro Miltersen, Vinodchandran N. Variyam, and Osamu Watanabe. Superpolynomial versus half-exponential circuit size in the exponential hierarchy. In *Computing and Combinatorics Conference* (COCOON), pages 210–220, 1999. 221

[138] Hiroki Morizumi. Limiting negations in formulas. In *International Colloquium on Automata, Languages and Programming* (ICALP), pages 701–712, 2009. 100, 120

[139] Hiroki Morizumi. Limiting negations in non-deterministic circuits. *Theor. Comput. Sci.*, 410(38-40):3988–3994, 2009. 100, 120

[140] Elchanan Mossel and Ryan O'Donnell. On the noise sensitivity of monotone functions. *Random Struct. Algorithms*, 23(3):333–350, 2003. 109, 111, 115, 116

[141] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. 127

[142] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004. 190

[143] Eduard I. Nečiporuk. On a Boolean function. *Soviet Math. Dokl.*, 7(4):999–1000, 1966. 52

[144] Ryan O'Donnell and Rocco A. Servedio. Learning monotone decision trees in polynomial time. *SIAM J. Comput.*, 37(3):827–844, 2007. 98

[145] Ryan O'Donnell and Karl Wimmer. KKL, Kruskal-Katona, and monotone nets. *SIAM J. Comput.*, 42(6):2375–2399, 2013. 98, 120

[146] Ryan O'Donnell. *Computational applications of noise sensitivity*. PhD thesis, MIT, 2003. 109

[147] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 12, 136, 140, 144

[148] Igor C. Oliveira. Algorithms versus circuit lower bounds. *Electronic Colloquium on Computational Complexity* (ECCC), 13:117, 2013. 51

[149] Janos Pach and Pankaj Agrawal. *Combinatorial Geometry*. Wiley-Interscience, 1995. 173

[150] Christos H. Papadimitriou and Michael Sipser. Communication complexity. *J. Comput. Syst. Sci.*, 28(2):260–269, 1984. 58

[151] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. *Chicago J. Theor. Comput. Sci.*, 1999. 187

[152] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. 14, 19, 20

[153] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992. 14, 98, 146

[154] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997. 10, 50, 185, 189, 190, 200

[155] Alexander A. Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical Notes*, 37(6):485–493, 1985. 5, 50

[156] Alexander A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR*, 281:798–801, 1985. English translation in: Soviet Mathematics Doklady 31:354–357, 1985. 5, 98

[157] Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987. 52, 53, 66, 71, 72, 92, 95

[158] Alexander A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya: Mathematics*, 59(1):205–227, 1995. 189

[159] Neil Robertson and Paul D. Seymour. Graph minors V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986. 46

[160] Benjamin Rossman. *Average-case complexity of detecting cliques*. PhD thesis, MIT, 2010. 202

[161] Benjamin Rossman. The monotone complexity of k-clique on random graphs. *SIAM J. Comput.*, 43(1):256–279, 2014. 5

[162] Benjamin Rossman. Correlation bounds against monotone NC^1. In *Conference on Computational Complexity* (CCC), 2015. 100

[163] Steven Rudich and Leonard Berman. Optimal circuits and transitive automorphism groups. In *International Colloquium on Automata, Languages and Programming* (ICALP), pages 516–524, 1988. 91

[164] Steven Rudich. Super-bits, demi-bits, and NP/qpoly-natural proofs. In *Workshop on Randomization and Computation* (RANDOM), pages 85–93, 1997. 190

[165] Miklos Santha and Christopher B. Wilson. Limiting negations in constant depth circuits. *SIAM J. Comput.*, 22(2):294–302, 1993. 100, 147

[166] Rahul Santhanam and Ryan Williams. Uniform circuits, lower bounds, and QBF algorithms. *Electronic Colloquium on Computational Complexity* (ECCC), 19:59, 2012. 189

[167] Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *Symposium on Foundations of Computer Science* (FOCS), pages 183–192, 2010. 187

[168] Rahul Santhanam. Ironic complicity: Satisfiability algorithms and circuit lower bounds. *Bulletin of the* EATCS, 106:31–52, 2012. 51, 186

[169] Dierk Schleicher and Malte Lackmann. *An Invitation to Mathematics: From Competitions to Research*. Springer, 2011. 1

[170] Stefan Schneider. Satisfiability algorithms for restricted circuit classes. *CoRR*, abs/1306.4029, 2013. 187

[171] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978. 204

[172] Rocco Servedio. On learning monotone DNF under product distributions. *Information and Computation*, 193(1):57–74, 2004. 98

[173] Kazuhisa Seto and Suguru Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. *Computational Complexity*, 22(2):245–274, 2013. 187

[174] Claude Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(1):59–98, 1949. 5

[175] Michael Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997. 2

[176] D. Sivakumar. Algorithmic derandomization via complexity theory. In *Symposium on Theory of Computing* (STOC), pages 619–626, 2002. 173

[177] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Symposium on Theory of Computing* (STOC), pages 77–82, 1987. 52, 53, 66, 71, 92, 95

[178] Philip M. Spira. On time-hardware complexity tradeoffs for Boolean functions. In *Hawaii International Conference on System Sciences* (HICSS), pages 525–527, 1971. 17

[179] Srikanth Srinivasan. On improved degree lower bounds for polynomial approximation. In *Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 201–212, 2013. 53, 54

[180] Shao Chin Sung and Keisuke Tanaka. An exponential gap with the removal of one negation gate. *Inf. Process. Lett.*, 82(3):155–157, 2002. 120

[181] Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: An extension of Markov's theorem. *Inf. Process. Lett.*, 90(1):15–20, 2004. 100, 120

[182] Michel Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996. 109, 131

[183] Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988. 26, 120

[184] Jun Tarui. Smallest formulas for the parity of 2^k variables are essentially unique. *Theor. Comput. Sci.*, 411(26-28):2623–2627, 2010. 91

[185] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. 216

[186] Iannis Tourlakis. Time-space tradeoffs for SAT on nonuniform machines. *J. Comput. Syst. Sci.*, 63(2):268–287, 2001. 203

[187] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. 156, 169, 172

[188] Christopher Umans. Pseudo-random generators for all hardnesses. *J. Comput. Syst. Sci.*, 67(2):419–440, 2003. 191, 221

[189] Salil P. Vadhan. Personal communication, 2012. 169

[190] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Symposium on Mathematical Foundations of Computer Science* (MFCS), pages 162–176, 1977. 51

[191] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. 216

[192] Leslie G. Valiant. Exponential lower bounds for restricted monotone circuits. In *Symposium on Theory of Computing* (STOC), pages 110–117, 1983. 51

[193] Leslie G. Valiant. A theory of the learnable. In *Symposium on Theory of Computing* (STOC), pages 436–445, 1984. 152, 216

[194] Leslie G. Valiant. Evolvability. *J. ACM*, 56(1), 2009. 157, 161

[195] Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009. 51, 54, 57

[196] Ilya Volkovich. On learning, lower bounds and (un)keeping promises. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 1027–1038, 2014. 181, 192

[197] Ingo Wegener. Relating monotone formula size and monotone depth of Boolean functions. *Inf. Process. Lett.*, 16(1):41–42, 1983. 17

[198] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Symposium on Theory of Computing* (STOC), pages 231–240, 2010. 10, 180, 186, 187, 188, 193, 202, 203

[199] Ryan Williams. Non-uniform ACC circuit lower bounds. In *Conference on Computational Complexity* (CCC), pages 115–125, 2011. 10, 180, 186, 188, 189, 193, 194, 201, 202, 203, 212

[200] Ryan Williams. Private communication, 2013. 188

[201] Ryan Williams. Natural proofs versus derandomization. In *Symposium on Theory of Computing* (STOC), pages 21–30, 2013. 10, 186, 188, 190, 195, 198, 200, 207, 211, 212, 220

[202] Ryan Williams. Algorithms for circuits and circuits for algorithms. In *Conference on Computational Complexity* (CCC), pages 248–261, 2014. 51

[203] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Symposium on Theory of Computing* (STOC), pages 664–673, 2014. 51

[204] Ryan Williams. Nonuniform ACC circuit lower bounds. *J. ACM*, 61(1):2, 2014. 51

[205] Andrew Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *Symposium on Foundations of Computer Science* (FOCS), pages 1–10, 1985. 50, 189

[206] Stanislav Žák. A Turing machine time hierarchy. *Theor. Comput. Sci.*, 26(3):327–333, 1983. 204