A Dual Perspective on Computational Complexity

by

Brynmor Chapman

B.A., University of Oxford (2013)
M.S., Stanford University (2016)
M.A., University of Oxford (2017)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

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

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

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

A Dual Perspective on Computational Complexity

by

Brynmor Chapman

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

Abstract

One central aim of theoretical computer science is to understand the limits of computation. On the one hand, algorithms give upper bounds on the resources required for a computation. They are intuitive, well-studied, and well-understood. On the other hand, proofs of hardness, or lower bounds as they are often called, seem to be elusive and relatively poorly understood. Perhaps the most famous conjectured lower bound is $\mathsf{P} \neq \mathsf{NP}$, which informally states that there are problems whose solutions can be efficiently verified but not efficiently computed. $\mathsf{P} \neq \mathsf{NP}$ is so widely believed that the world economy relies critically on its truth. Indeed, complexity theorists generally believe the much stronger Exponential Time Hypothesis (ETH). However, despite decades of work, these conjectures remain open, as do much weaker conjectures such as $\mathsf{P} \neq \mathsf{PSPACE}$.

The aforementioned discrepancy between what is known and what is believed is fairly representative of the state of the art in complexity theory, which invites the question: is it inherent? Are lower bounds somehow inherently harder to prove than upper bounds? There is a specific, provable sense in which the answer is yes. There are indeed proof barriers, which show that broad classes of proofs have no hope of proving many of the commonly studied lower bounds. These barriers have led to a fair amount of pessimism in complexity theory. The aim of this thesis is to study lower bounds from an algorithmic perspective, in order to use well-studied algorithmic techniques to advance complexity theory. It covers three facets of this perspective. It presents a way to bypass the Natural Proofs Barrier of Razborov and Rudich, discusses dualities between upper and lower bounds, and disproves longstanding conjectured lower bounds.

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

Acknowledgements

I would like to thank my family, friends, and coworkers who directly and indirectly made this work possible. I'd especially like to thank:

Ryan Williams, my advisor, for being such a brilliant mentor and friend over the years, for your wisdom, kindness, quite frankly legendary patience, and tolerance of (complicity in?) all of my shenanigans. You not only shared in the good times and made MIT (and even Stanford) so much more inviting, but you also helped me through the bad times and didn't give up on me even when you had every right to.

The Chap-people, including my parents, Michael and Carolyn, my brothers, Owen, Evan, and Dewi, and our new little "sister", Aiko, for your constant love and support, for making me who I am now, for all of the sacrifices (willing and otherwise) that you've made along the way, and for trying to make sure I eat properly. Aiko, keep taking good care of Dewi, and Dewi, try not to let Aiko onto rooftops.

The rest of my family, including Helen Chapman, Louise, Frances, and Edward Norris, Hisako, Shiro, Debbie, and Shota Ogushi, Haru and Scott Nakasone, and Peter Kohama, for always making time for each other and me, for tolerating the ensuing chaos, and for encouraging me on this journey.

Alex "Quadling" Lombardi for your constant companionship since I arrived at MIT, for feeding me, for all the time spent chattering about crypto, complexity, and quantum, for all the time wasted on Pokémon, cards, crosswords, Zelda, League of Legends, anime, just getting lost, and plenty of other things besides, and for all the time somehow both spent productively and wasted when I napped through you chattering to someone else.

Jess "Kitty" Xu for joining, redirecting, dialing up, and chronicling the above shenaningans, for adding head pats, cuddles, and meaps to the mix, for feeding me (perhaps less intentionally then Quadling does - sorry about that...), for taking me under your wing at Sciarappa, and for teaching me what a good relationship can be like.

Luke "Grandpa" Schaeffer for quantum and Euro-theory discussions, crosswords,

cards, and somehow always knowing the right thing to say even though it would usually be totally wrong coming from anyone else.

Eric Lu and Jiwon "Yuna-unnie"[1] Joung for your invaluable contributions to the Sciarappa degeneracy, for your green thumb and baking expertise, and for trying to teach me a little bit about systems.

Vy "Aya" Nguyen, Yota "Nezumi" Kato, and Kevin Chen for help with my minor and for the cards and other games, especially at the beginning of the pandemic. Vy, thank you also for welcoming me into the Pokémon community, for a sympathetic ear, a shoulder to cry on, and a place to crash when they were needed, and for all of the desserts.

Virginia and GP Williams for Music Night with Ryan, joint group research lunches, holiday parties, and random discussions about graphs, including a particularly interesting homework problem that we ended up giving to our own students.

Samanta, Natalia, Kamila, and Olek Mądry for bringing so much joy and laughter to the Theory Group and Music Night, and for sharing Warszawa with me.

Andrea Leang and Yichen Gao for reminding me of the joys of teaching after the struggles with the pandemic and Zoom University, and for supporting my quest to stay at MIT. Andrea, thank you also for being so gentle and understanding regarding my rather peculiar difficulty involving masks, and for encouraging me to take up my sword once more.

Dylan McKay for your insights about teaching, for being a keystone of Music Night and Theory Tea, and for rolling with everything Ryan and I threw your way.

Carolyn Kim for support, company, and a nice work environment, particularly while putting together [31] during my last months at Stanford.

Karlanna Lewis for company and a place to stay during SoCal conferences. Nuri, thank you also for tolerating me, even in the new apartment.

In memoriam Eric Chapman, Jean Chapman, and Hisashi Ogushi, my grandparents who are unable to see this completed work. You are dearly loved and dearly missed.

[1] Yes, I know.

Contents

Chapter 1

Introduction

1.1 Motivation

One of the central aims of theoretical computer science is to understand the power of computation, in terms of one or more of a number of different resources, e.g.,

- number of operations required,
- time required by some number of parties operating in parallel,
- amount of communication between two or more cooperating parties,
- memory use,
- number of random bits used,
- number of quantum bits used,
- error tolerance,
- number of passes over a large input,
- or size/depth of a Boolean circuit that performs the desired computation.

Algorithms, or *upper bounds* as they are often called, state that computation is "easy" in some way. Upper bounds are ubiquitous in computer science and indeed in

daily life. Uses include the obvious personal computer, as well as perhaps less obvious uses such as a watch, key, credit card, or automobile. They are relatively well-studied and understood. People seem to have an intuitive understanding for algorithms, as evinced by small children, who naturally use algorithms for tasks such as sorting, arithmetic, and various puzzles. Indeed, in some cases, they may intuit or even rediscover asymptotically optimal algorithms like Sir Tony Hoare's QuickSort [63].

In contrast to upper bounds, hardness results, or *lower bounds*, state that computation is "hard" in some way. Perhaps the most famous lower bound (and indeed perhaps the most famous open problem in computer science) is P versus NP. Informally, are all problems with easily verifiable solutions also easy to solve? This problem was first stated precisely circa 1971 by Stephen Cook [41] and Leonid Levin [70], although it had been studied earlier in various forms [79, 73]. Cook and Levin pioneered the study of NP-completeness, which describes problems that are no easier than any NP problem, in the sense that an efficient solution to any NP-complete problem yields efficient solutions to all NP problems. The canonical example of an NP-complete problem is the Boolean Satisfiability Problem, or SAT. This is the problem of deciding whether a Boolean formula has any satisfying assignment. It is generally believed that there are problems (e.g., SAT) that are difficult to solve but have easily verifiable solutions. The conjectured lower bound is stated as follows:

Conjecture 1.1.1 ($\mathsf{P} \neq \mathsf{NP}$) *SAT cannot be solved in time polynomial in its input length, i.e., in time* $n^{O(1)}$.

Complexity theorists are collectively so confident in this conjecture that the world economy relies essentially on its truth. If it were to be disproven, then all secure communication, including all electronic commerce and banking, would be compromised. In fact, much stronger lower bounds are also widely believed. One example of such a conjectured lower bound is the Exponential Time Hypothesis, or ETH. ETH deals with a restricted variant of SAT called 3-SAT, in which each clause of the input formula may have at most three literals. Although ostensibly easier than SAT, 3-SAT is also known to be NP-complete.

Conjecture 1.1.2 (ETH) *3-SAT cannot be solved in time subexponential in the number of variables, i.e., in time* $2^{o(n)}$.

On the other hand, despite mathematicians and computer scientists working on P versus NP in some form for over sixty years, even much weaker lower bounds remain elusive. Where SAT is the problem of determining the validity of a quantified Boolean formula with only existential quantifiers, the Quantified Boolean Formula Problem (QBF) is the problem of determining the validity of such a formula when both existential and universal quantifiers are allowed. Ostensibly, QBF is much harder than SAT, and yet the following is still an open problem:

Conjecture 1.1.3 *QBF cannot be solved in time linear in its input length, i.e., in time* $O(n)$.

The above discrepancy between what is known and what is believed is fairly representative of the state of the art in complexity theory. Optimal or near-optimal upper bounds are often proven before matching lower bounds, and where there is a large gap between known upper and lower bounds, the upper bounds are often believed to be much closer to optimal. Are lower bounds inherently harder to prove than upper bounds? While some may argue the affirmative, the aim of this thesis is to study lower bounds from an algorithmic perspective, in order to use well-studied algorithmic techniques to advance complexity theory. Chapters 3-7 present three facets of this perspective. Chapter 3 shows how to circumvent one of the proof barriers that have led to pessimism in complexity theory. Chapters 4 and 5 discuss dualities between upper and lower bounds. Chapters 6 and 7 refute longstanding conjectured lower bounds.

1.2 The Circuit Model

This thesis focuses on the *circuit model* of computation. In contrast to an algorithm in the classical model, which handles every possible input, a circuit can be viewed as an algorithm that only takes inputs of one particular length. An "algorithm" in the

circuit model then consists of an infinite family of circuits, one for each possible input length. Note that this model is more powerful than the classical model. Even families of trivial circuits can compute undecidable functions, which cannot be computed classically. Circuit complexity is a promising avenue to resolving some of the open problems in complexity theory. To prove $\mathsf{P} \neq \mathsf{NP}$, it would suffice to prove that NP is not contained in the circuit class $\mathsf{P/poly}$, which consists of functions computable by polynomial size circuit families. For more detail on the circuit model of computation and circuit classes including $\mathsf{P/poly}$, see Chapter 2.

Chapters 3 and 4 consider the following zero-sum game studied in prior work (e.g., [74, 45]). Fix a Boolean function f. A *circuit player* chooses from a set of Boolean circuits, while an *input player* chooses from a set of inputs. The circuit player wins if the chosen circuit computes f on the chosen input; otherwise, the input player wins.

The circuit-input game was first explicitly studied by Lipton and Young [74], in the context of providing complexity-theoretic applications of strategies for general zero-sum games (see also [106]). Among other results, they (and independently, Newman [81] and Althöfer [7]) proved that approximate and succinct strategies exist for any zero-sum game in the sense for an $m \times n$ matrix M, there exists a strategy for the row player with support size $O(\log n)$, and a strategy for the column player with support size $O(\log m)$, which together additively approximate the optimal strategy of the game. This result was used to prove the existence of so-called *anti-checkers*: for any function $f : \{0,1\}^n \to \{0,1\}$ with circuit complexity at least s, there exists a set S of $O(s)$ n-bit inputs on which all circuits of size at most s/n fail to compute f correctly on a $1/2 - \varepsilon$ fraction of inputs in S.

Chapter 2 concludes with a proof of a general extension of these classical results, along with a proof that this extension cannot be improved in a certain technical way. This extension is then used in Chapters 3 and 4 to present two new applications of these classical results to the field of circuit complexity.

1.3 Natural Proofs

Chapter 3 gives an alternative view into the Natural Proofs Barrier of Alexander Razborov and Steven Rudich [91] and in particular suggests a new pathway around it. Natural proofs have three properties.

- They are *constructive*: they contain an efficient algorithm A from a complexity class Γ for testing Boolean functions given as truth tables,
- they are *large*: algorithm A accepts a large fraction (at least $1/2^{O(n)}$) of all n-bit Boolean functions, and
- They are *useful*: algorithm A rejects all functions which are truth tables of circuits from a circuit class $\mathcal{C}$, for infinitely many input lengths.

Such an algorithm A is called a Γ-*natural property useful against* $\mathcal{C}$, and Razborov-Rudich showed any P/poly-natural property useful against P/poly would preclude the existence of the cryptographic primitive known as a *pseudorandom function*. A pseudorandom function is a function that is easy to compute, but whose output looks like random noise. Pseudorandom functions are essential in cryptography and believed to exist. Hence a natural proof should not be capable of proving P/poly lower bounds, or statements like $\mathsf{NP} \not\subset \mathsf{P}$/poly. This ruled out many potential methods for proving circuit lower bounds.

A minor (and in hindsight, obvious) modification to the "useful" condition of natural proofs not only makes the barrier disappear, but it makes circuit lower bounds equivalent to the existence of such modified natural properties. This minor modification is perhaps best illustrated by considering NP vs P/poly and the SAT problem (although any self-reducible NP-complete problem would suffice). It begins with the well-known observation that any polynomial-size circuit C can be assumed, without loss of generality, never to err when it reports satisfiability. That is, given a circuit C that potentially solves SAT, C can be augmented to generate a satisfying assignment: create a larger circuit C' that contains copies of C, such that when C reports "SAT" on a formula, C' repeatedly plugs in values for variables of the formula and queries C

on the reduced formulas to check if satisfiability still holds. Either C will eventually be in error (in which case C' reports "UNSAT") or C will produce a SAT assignment, in which case C' reports "SAT" without error.

Let C be a Boolean circuit on n inputs. Define C to be a *SAT solver* if for all n-bit formulas φ, $C(\varphi) = 1$ implies that φ is satisfiable. (Such circuits are called "SAT solvers" because one could use these circuits to print satisfying assignments to formulas, when the circuits report "SAT.") The class of functions computable by polynomial-size SAT solvers is an expressive class, including special cases such as 2-SAT, Horn-SAT, etc. By the previous paragraph, to prove $\mathsf{NP} \not\subset \mathsf{P/poly}$ it suffices to prove that no polynomial-size family of SAT solvers can compute SAT. That is, a lower bound proof only needs to be *useful* against small SAT solvers. But *how* might one prove this? By using combinatorial or probabilistic methods, one might expect to find an efficient test for Boolean functions, such that random functions (and SAT) pass, but SAT solver circuits do not pass. This looks very much like a natural proof, except instead of rejecting all polynomial-size circuits, the test will only try to reject the polynomial-size SAT solvers. It turns out that such tests must exist, if $\mathsf{NP} \not\subset \mathsf{P/poly}$.

Theorem 1.3.1 (Informal, cf. Theorem 3.2.1) $\mathsf{NP} \not\subset \mathsf{P/poly}$ *if and only if there is a natural property that is useful against polynomial-size SAT solvers and accepts SAT.*

Compare with the main theorem of Razborov-Rudich: if there are $\mathsf{P/poly}$-natural properties useful against $\mathsf{P/poly}$, then there are no secure pseudorandom functions. The above theorem suggests that, to circumvent "naturalness" and prove circuit lower bounds for SAT, it would be prudent to try to look for proof methods which *fail* on arbitrary polynomial-size circuits, but *succeed* on circuits that try to print full satisfying assignments. This point also gives intuition for why Theorem 3.2.1 holds without hurting cryptography: the truth tables of SAT solvers do not look at all like random functions, so a natural property useful against SAT solvers is in no danger of

distinguishing pseudorandom function candidates from truly random functions.[1]

Equivalences similar to Theorem 3.2.1 hold for other circuit lower bound problems. In general, for any function f that permits a zero-error or one-sided error corrector in a complexity class $\mathcal{C}$, there is an equivalence between proving $f \notin \mathcal{C}$ and exhibiting natural properties useful against "error-corrected f-solving circuits." Chapter 3 also shows how a version of the statement holds for f with (randomized) polynomial-time program checkers [24].

1.4 Testing Circuits for Functionality Using Data

Chapter 4 applies succinct strategies for zero-sum games to set up a framework that is "dual" to the usual computational view of circuits computing functions on inputs, treating inputs as the "programs" and circuits as the "input data". This generalizes Lipton-Young's anti-checkers, showing how the general problem of circuit lower bounds can be seen in a constructive light: as designing small data sets that can be used to conclusively test whether a given circuit computes a particular function. It also yields a complexity-theoretic perspective on the practical problem of software testing (see [89, 101]).

Consider a language L, called the *primal* language. The primal complexity measure is then the standard measure of the circuit complexity of L, i.e. the smallest number of gates required by a circuit that computes membership in L. (Note that this is a function of the number of input bits.)

The *dual* problem, TEST-L, is then the set of *circuits* that are consistent with L, in the sense that they compute L correctly on all inputs of the relevant length. TEST-L can be computed with a test suite of inputs by checking to see that a given circuit computes L correctly on all inputs in the test suite. Say that such a test suite computes TEST-L correctly if every circuit that is *not* consistent with L errs on at least one input in the test suite. The dual complexity measure, or *data complexity*,

[1] As was succinctly noted, one might interpret this result as saying, "pseudorandom functions have no hope of computing SAT, so why should we care about them?"

is then the smallest number of inputs required by a test suite to compute TEST-L. Note that the data complexity is a function of the size of the *circuit* being tested, as this measures the "length of the input" to the problem TEST-L. Every language L can be computed by trivial circuits of size exponential in the input length. However, the theory of circuits becomes interesting when only small circuits are considered. Similarly, there are also always trivial test suites of size exponential in the size of the circuits being tested that can conclusively determine whether a circuit correctly computes L. It would suffice to try all possible inputs of size up to the size of the circuit being tested. However, a natural goal is to design smaller test suites that still compute TEST-L correctly.

The main theorem of Chapter 4 uses the circuit-input game in a crucial way to prove a strong correspondence between the primal and dual complexity measures of L, one consequence of which is the following:

Theorem 1.4.1 (Informal, cf. Corollary 4.2.3) $\mathsf{NP} \not\subset \mathsf{P}/poly$ *if and only if SAT has subexponential data complexity.*

The circuit lower bound $\mathsf{NP} \not\subset \mathsf{P}/\mathrm{poly}$ can then be thought of algorithmically, as the problem of designing efficient test suites.

1.5 Black-Box Hypotheses

Chapter 5 studies obfuscation from a different direction compared to most existing literature on the subject. It proposes generalized forms of the "Black-Box Hypothesis" considered in Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [14] and relates them to lower bound questions.

In the pioneering work of Barak *et al.* [14] on obfuscation, the authors also proposed a compelling conjecture about black-box obfuscation that they called a "Scaled-Down Rice's Theorem" [14, Conjecture 5.1]; the conjecture has recently been renamed the *Black Box Hypothesis* (BBH) [93, 65]. Informally, the Black-Box Hypothesis posits that, when code is represented as a small Boolean circuit, and a code analyst is

represented as an efficient algorithm, the only possible analysis tasks are ones that could have been performed using only the input-output behavior of the code (and not the code itself).

The original Black-Box Hypothesis is still a major open problem, but other natural variants of the hypothesis may be more tractable to resolve *unconditionally*. Chapter 5 considers variants of the Black-Box Hypothesis in a more general complexity-theoretic setting, where the complexity of the analyst, the complexity of the code being analyzed, and the function to be obfuscated (the "box") are taken carefully into account. For example, of particular interest are the cases where the "analyst function" is taken from a "low" complexity class $\mathcal{A}$ (smaller than P), and the box is also from a "low" complexity class $\mathcal{C}$.

Hypothesis 1.5.1 ($\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$) [Informal, cf. Hypothesis 5.4.1] *The only analysis tasks that can be performed by analysts from $\mathcal{A}$ on circuits from $\mathcal{C}$ are ones that could have been performed using only black-box oracle access to the analyzed circuit.*

In prior work, the class of analysts $\mathcal{A}$ was always set to be BPP, and the class of circuits $\mathcal{C}$ was generally set to be unrestricted circuits of fan-in two. In that full form, proving the BBH would also prove $\mathsf{NP} \not\subseteq \mathsf{BPP}$, and hence also $\mathsf{P} \neq \mathsf{NP}$, so that is presently out of reach! (The BBH could also end up being false, of course.) By considering a range of natural possible choices for the weak analysts $\mathcal{A}$ and the circuit sets $\mathcal{C}$, it may be possible to delineate precisely how weak the analysts from $\mathcal{A}$ need to be, in order for $\mathcal{C}$-circuits to *provably* behave like black boxes, and to relate the corresponding Black-Box Hypotheses to other core problems within complexity.

Chapter 5 demonstrates several interesting relationships between circuit lower bounds and Black-Box Hypotheses in the generalized setting. First, it uses known circuit lower bounds to prove that certain instances of the Black-Box Hypothesis are true. In fact, it presents a generic connection between lower bounds and Black-Box Hypotheses. It also presents some converse results, showing that Black-Box Hypotheses imply certain circuit lower bounds. Finally, in some settings, it can be shown that

certain problems are "complete" for a Black-Box Hypothesis, in the sense that proving the Black-Box Hypothesis is *equivalent* to proving a lower bound against the aforementioned problem.

1.6 Gotsman-Linial Conjecture

Chapter 6 studies a different complexity measure of Boolean functions. A Boolean function is an *n-variate, degree-d Polynomial Threshold Function*, or (n, d)-PTF, if it can be expressed as the sign of a (real) polynomial of degree at most d evaluated on the n-dimensional Boolean hypercube. This definition alone is not terribly exciting without restrictions on d, as every Boolean function on n variables can be written as the sign of (and in fact can be written exactly as) a multilinear polynomial of degree n. Of particular interest is the case where d is small.

In an influential paper, Craig Gotsman and Nathan Linial [53] applied Fourier analytic techniques to the study of PTFs. They were mainly interested in connecting different measures of the complexity of Boolean functions, and of low-degree PTFs in particular. One such measure was the Average Sensitivity of a Boolean function, which is a normalized measure of how often changing a single input bit changes the output value of the function.

Among other things, Gotsman and Linial proved a tight upper bound on the average sensitivity of linear threshold functions, or $(n, 1)$-PTFs, achieved by the MAJORITY function on n variables. They conjectured that this bound generalizes to higher degree PTFs, in that the (n, d)-PTF of maximal average sensitivity is the obvious symmetric candidate, which alternates signs nearest where half of the inputs are ones.

Conjecture 1.6.1 (Gotsman-Linial, informal, cf. Conjecture 6.1.1) *For every n and d, at least one (n, d)-PTF of maximal average sensitivity is symmetric.*

This conjecture was listed as a prominent open problem in [83] and [44]. If true, it would have many applications in complexity and learning (see for example [60, 52, 67, 69, 38]). Gotsman and Linial proved their conjecture for the case where $d = 1$,

and it is also known to be true in the case where $d = 0$. However, it was left open whether the conjecture holds for any $d \geq 2$.

Chapter 6 resolves the Gotsman-Linial Conjecture for all pairs (n, d) except the case when $n > 7$ is even and $d = 2$. The main results are the following:

Theorem 1.6.1 (Informal, cf. Theorem 6.1.1) *For all pairs (n, d) of natural numbers satisfying one of the following criteria, there exists an (n, d)-PTF witnessing a counterexample to the Gotsman-Linial Conjecture:*

- *$n \geq 5$ is odd, and $d = 2$.*
- *$n \geq 7$, and $3 \leq d \leq n - 3$.*

Theorem 1.6.2 (Informal, cf. Theorem 6.1.2) *For all pairs (n, d) of natural numbers satisfying one of the following criteria, a symmetric function has the greatest average sensitivity among (n, d)-PTFs:*

- *$d \leq 1$.*
- *$d \geq n - 2$.*
- *$n = 6$.*

The results (and the remaining open cases) are summarized in Figure 1-1.

1.7 Counting Circuits for Symmetric Functions

Chapter 7 studies constant-depth circuits in which every (unbounded fan-in) gate (called a MOD_m gate) determines whether the sum of its inputs is divisible by a small constant integer m. Although the model looks rather peculiar, constant-depth circuits with constant moduli gates (a.k.a. CC^0 circuits, a.k.a. pure-ACC circuits [105]) have been a longstanding and fundamental roadblock in the way of improved circuit complexity lower bounds. Since their identification over 30 years ago [15, 19], scant progress has been made on lower bounds against CC^0, and its close cousin ACC^0 which

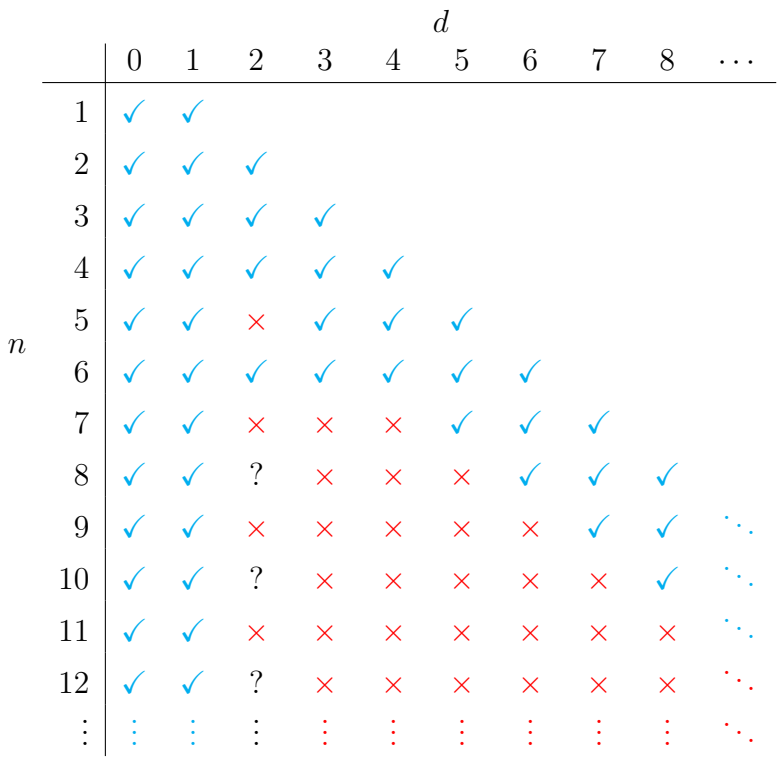


Figure 1-1: Results are summarized in the above table. A cyan tick mark indicates a case in which the conjecture holds (for all (n,d)-PTFs f, $\mathbf{AS}[f] \leq \mathbf{AS}[f^*_{n,d}]$). A red cross indicates a refutation (there exists an (n,d)-PTF f such that $\mathbf{AS}[f] \in (1+\Omega(n^{-1}e^{-d^2/n}))\mathbf{AS}[f^*_{n,d}]$). A black question mark indicates an open case. Note: the cases $(n,d)=(6,2)$ and $(n,d)=(6,3)$ were verified with the help of a computer search and a linear program solver (see Appendix 6.5).

includes AND and OR in the gate basis. Some exceptions include work focusing on special cases of the problem (e.g., [16, 55, 35, 36]), uniform lower bounds [3], and work proving strong lower bounds but only for functions whose complexity is in QuasiNP or higher (e.g., [103, 39, 78, 37]). If there has ever been a "circuit complexity winter", CC^0 circuits are at least partly to blame.

Besides simple ignorance, could there be deeper reasons why CC^0 circuits have been so difficult for showing limitations? Chapter 7 explores the possibility that CC^0 circuits may be powerful, focusing on the natural class of *symmetric Boolean functions* whose output depends only on the number of ones in the input. It has been conjectured for many years that the AND function does not have polynomial-size CC^0

circuits ([17, 100, 99])[2]. It is well-known that AC^0 circuits, which consist of AND, OR, and NOT gates and have constant depth, require exponential size to compute arbitrary symmetric functions [61]. In recent work, Oliveira, Santhanam, and Srinivasan [85] have shown that PARITY gates (a.k.a. MOD_2 gates) can help compute symmetric functions more efficiently than what AND, OR, NOT can accomplish in constant depth, although it should be noted that these upper bounds are still exponential. This prompts the question: what about other moduli? Chapter 7 shows that low-depth MOD_m circuits with arbitrary but fixed modulus m can actually compute arbitrary symmetric Boolean functions (such as MAJORITY) much more efficiently than low-depth circuits with AND, OR, and MOD_q gates, when q is a prime power.

Theorem 1.7.1 (Informal, cf. Theorem 7.1.1) *Every symmetric function can be computed with depth-3* CC^0 *circuits of subexponential size.*

That is, without any AND/OR gates, it is possible to obtain CC^0 circuits with a substantially smaller number of gates than the longstanding lower bounds for $\mathsf{AC}^0[q]$ circuits computing symmetric functions (mentioned in the previous paragraph), for prime power q. Chapter 7 also includes generalizations of the above upper bound, giving size-depth tradeoffs and extensions to ACC^0. It concludes with a discussion of a natural conjecture under which these upper bounds cannot be improved substantially.

1.8 Sources

This work was previously published in the following sources:

1. The circuit-input game, natural proofs, and testing circuits with data
   Chapman & Williams, ITCS 2015 [32]

2. The Gotsman-Linial Conjecture is false
   Chapman, SODA 2018 [31]

[2]Hansen and Koucký [58] give an interesting counterpoint, showing that *probabilistic* CC^0 circuits can compute AND efficiently. Thus the AND $\in \mathsf{CC}^0$ problem is equivalent to a derandomization question.

3. Black-box hypotheses and lower bounds
   Chapman & Williams, MFCS 2021 [33]

4. Smaller ACC^0 circuits for symmetric functions
   Chapman & Williams, ITCS 2022 [34]

Chapter 2

Preliminaries

2.1 Background

This thesis assumes familiarity with computational complexity and the circuit model of computation in particular. Knowledge of the first thirteen chapters of Arora and Barak [10] should suffice. (For the reader unfamiliar with circuit complexity, the following section gives a brief overview of the required concepts.) Beyond this and where explicitly excepted, each chapter will be self-contained. Notation and definitions will be introduced or recalled as necessary. Sometimes, as is common in complexity theory, the distinction will be blurred between a complexity class $\mathcal{C}$ as a set of decision *problems* and the set of *algorithms* that solve these problems.

2.2 Circuit Complexity

This section recalls the basics of circuit complexity. The knowledgeable reader may skip to Section 2.3 without consequence.

This thesis uses a model of computation called a *Boolean circuit*, which differs slightly from the classical Turing Machine. In the classical model, a Turing Machine must handle all possible inputs of all possible lengths. In contrast, a Boolean circuit only takes inputs of one particular length. It has a natural interpretation as an ab-

straction of e.g. a CPU, which might only take 64-bit inputs. Besides being a natural model of computation, Boolean circuits are also mathematically simpler than Turing Machines. Hence since the 1980s, mathematicians and complexity theorists have considered proving lower bounds against Boolean circuits as a promising approach towards resolving classical complexity questions like P vs. NP.

In the following, let $\mathcal{B}$ be a set, or *basis*, of Boolean functions.[1] When the basis is obvious or irrelevant, it is often omitted.

Definition 2.2.1 (Circuit) *A* Boolean circuit *C over basis $\mathcal{B}$ on n inputs $x_1, \ldots, x_n$ is a directed acyclic graph in which the vertices are called* gates *and the edges are called* wires*. Exactly n gates, called* input gates*, have in-degree* 0 *and are labeled $x_1, \ldots, x_n$. All other gates are labeled with a function from $\mathcal{B}$. If a gate g is labeled with a k-ary function, then it must have exactly k (ordered) input wires. The* output gate *has out-degree* 0*.*

A Boolean circuit is said to compute a Boolean function in a fairly natural way.

Definition 2.2.2 *The* function computed by *C, denoted $C(\mathbf{x})$, is the n-ary function defined as follows. Let G be the set of gates in C. Let $\phi : G \times \{0,1\}^n \to \{0,1\}$ be defined inductively according to the topological order of G by:*

- *If the gate g is an input gate with label x_i, then $\phi(g, \mathbf{x}) = x_i$.*
- *If g is labeled with the function f and has input wires from gates $g_1, \ldots, g_k$, then $\phi(g, \mathbf{x}) = f(\phi(g_1, \mathbf{x}), \ldots, \phi(g_k, \mathbf{x}))$.*

Define $C(\mathbf{x}) = \phi(g^*, \mathbf{x})$, where g^* is the output gate of C.

Note that a circuit only computes a Boolean function on n bits, not a function on all binary strings. An "algorithm" in the circuit model is called a *circuit family*.

Definition 2.2.3 (Circuit Family) *A* circuit family *is a sequence $\{C_i\}_{i \in \mathbb{N}}$ of Boolean circuits, where C_i has i inputs.*

[1] Commonly studied bases include $\{\textsc{And}_2, \textsc{Or}_2, \textsc{Not}\}$, B_2 (the set of all binary Boolean functions), and U_2 (B_2 without Parity or its complement).

Definition 2.2.4 *The* function computed by $\{C_i\}$ *is the function* $C : \{0,1\}^* \to \{0,1\}$ *that extends the restrictions* $C_i : \{0,1\}^i \to \{0,1\}$ *computed by all of the* C_i*.*

Definition 2.2.5 *The* language accepted by $\{C_i\}$ *is the set of binary strings given by* $L(\{C_i\}) := \{s \mid C(s) = 1\}$*.*

The circuit model of computation gives rise to a natural complexity measure of Boolean functions, which roughly corresponds to time complexity in the classical model.

Definition 2.2.6 *The* size *of a circuit* C*, denoted* $|C|$*, is the number of gates in* C*.*

Definition 2.2.7 *The* size *of a circuit family* $\{C_i\}$ *is the function* $s : \mathbb{N} \to \mathbb{N}$ *given by* $s(n) = |C_n|$*.*

Definition 2.2.8 (SIZE(s)) *Let* $s : \mathbb{N} \to \mathbb{N}$*. The complexity class* SIZE(s) *is the set of all functions computable by circuit families of size dominated pointwise by* s*.*

And conversely:

Definition 2.2.9 (Circuit Complexity) *The* circuit complexity *of a function* $f : \{0,1\}^* \to \{0,1\}$ *is the size of the smallest circuit family computing* f*.*

In circuit complexity, the analogue of P is the complexity class P/poly, the set of all functions of polynomial circuit complexity.

Definition 2.2.10 (P/poly) $\mathsf{P}/poly := \bigcup_{c \in \mathbb{N}} \textsc{Size}(n \mapsto n^c + c)$.

The notation in the above definition is rather clumsy, so as is standard in complexity theory, the remainder of this thesis uses the following shorthand.

Notation 2.2.1 *For a function* $f : \mathbb{N} \to \mathbb{N}$*,* SIZE($f(n)$) *is taken to mean* SIZE(f)*.*

Notation 2.2.2 *For a set* S *of functions,* SIZE(S) *is taken to mean* $\bigcup_{f \in S} \textsc{Size}(f)$*.*

Example 2.2.1 $\mathsf{P}/poly = \bigcup_{c \in \mathbb{N}} \textsc{Size}(n^c + c) = \textsc{Size}(n^{O(1)})$.

One of the central questions of circuit complexity is whether P/poly contains all of NP. It is widely believed that $\mathsf{NP} \not\subset \mathsf{P}/\text{poly}$. This is a stronger statement than $\mathsf{P} \neq \mathsf{NP}$, since $\mathsf{P} \subset \mathsf{P}/\text{poly}$. To see this, observe that any polynomial time algorithm can be converted *uniformly*, or algorithmically, into a polynomial size circuit family that simulates it. However, P/poly also allows for non-uniform computation by circuit families that cannot be generated by any classical algorithm. For instance, P/poly contains every unary language, as each circuit in a family accepting such a language accepts at most one input. This includes unary encodings of undecidable languages such as the Halting Problem.

A second interpretation of P/poly and Boolean circuits in general is as classical algorithms that take *advice*. An advice string is a second auxiliary input that may change with the length of the primary input, but must remain constant for all primary inputs of the same length. A circuit family can be simulated by a classical algorithm for the Circuit Evaluation Problem Ckt-Eval, with the description of the correct circuit given as advice. Conversely, a classical algorithm that takes advice can be converted into a circuit family in which the advice strings are hard-coded. More general non-uniform complexity classes can be defined as follows:

Definition 2.2.11 *Let $\mathcal{C}$ be a (uniform) complexity class, and let $f : \mathbb{N} \to \mathbb{N}$. The complexity class $\mathcal{C}/f$ is the set of functions computable by algorithms in $\mathcal{C}$ when given advice strings of length $f(n)$ for inputs of length n.*

The concept of advice is useful for decoupling circuit complexity from the amount of non-uniformity (as a resource) required by a computation, particularly when the advice strings are small (sublinear in the length of the primary input). For example, P/poly contains all unary languages, as mentioned above, but so does the smaller complexity class $\mathsf{P}/1$.

Another natural complexity measure arising from Boolean circuits roughly corresponds to the time complexity of parallelized computation.

Definition 2.2.12 (Circuit Depth) *The* depth *of a circuit C is the number of wires on a longest (directed) path from an input gate to the output gate.*

Of particular importance to this thesis are constant depth circuits over several different bases. In all of the following, gates may have unbounded fan-in. By convention, NOT gates are always allowed and are "free", contributing to neither the size nor the depth of circuits.

Definition 2.2.13 (MOD Gate) *For an integer m, a MOD_m gate outputs 1 when the sum of its input bits is an integer multiple of m.*

Definition 2.2.14 (MAJ Gate) *A MAJ gate outputs 1 when a (non-strict) majority of its input bits are 1.*

Definition 2.2.15 (Alternating Circuits) *For an integer i, AC^i is the set of functions computable with circuits of $O(\log^i(n))$ depth and polynomial size over the basis $\{\text{AND}, \text{OR}\}$.*

Definition 2.2.16 (Counting Circuits) *For integers i, m, $\mathsf{CC}^i[m]$ is the set of functions computable with circuits of $O(\log^i(n))$ depth and polynomial size over the basis $\{\text{MOD}_m\}$.*

Definition 2.2.17 *For an integer i, $\mathsf{CC}^i := \bigcup_{m \in \mathbb{N}} \mathsf{CC}^i[m]$.*

Definition 2.2.18 *For integers i, m, $\mathsf{AC}^i[m]$ is the set of functions computable with circuits of $O(\log^i(n))$ depth and polynomial size over the basis $\{\text{AND}, \text{OR}, \text{MOD}_m\}$.*

Definition 2.2.19 (Alternating Circuits with Counting) *For an integer i, $\mathsf{ACC}^i := \bigcup_{m \in \mathbb{N}} \mathsf{AC}^i[m]$.*

Definition 2.2.20 (Threshold Circuits) *For an integer i, TC^i is the set of functions computable with circuits of $O(\log^i(n))$ depth and polynomial size over the basis $\{\text{MAJ}\}$.*

More specifically:

Notation 2.2.3 *For a constant d, AC^0_d (resp. $\mathsf{AC}^0_d[m]$, ACC^0_d, CC^0_d, TC^0_d) denotes the set of functions computable with AC^0 (resp. $\mathsf{AC}^0[m]$, ACC^0, CC^0, TC^0) circuit families of depth d.*

When the structure of a circuit is important, function composition notation will be used.

Example 2.2.2 MAJ $\circ$ AND $\circ$ OR *denotes TC^0_3 circuits in which all paths from an input gate to the output gate have length 3, the output gate is a* MAJ *gate, the layer of gates nearest the inputs are all* OR *gates, and the middle layer of gates are all* AND *gates.*

This will occasionally be extended to replace gate layers with circuit classes in the obvious way.

Example 2.2.3 MAJ $\circ \mathsf{AC}^0$ *denotes TC^0 circuits in which the output gate is a* MAJ *gate, and no other gate is a* MAJ *gate.*

Several chapters require explicit representations of Boolean functions. Unless otherwise specified, a Boolean function will be represented by its truth table, defined below.

Definition 2.2.21 (Truth Table) *The* truth table *of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is the 2^n bit string that enumerates the outputs of f on all n-bit inputs in lexicographical order.*

2.3 The Circuit-Input Game

The remainder of this chapter and the next two consider a zero-sum two-player game called the Circuit-Input Game, studied by Lipton, Young, Newman, and Althöfer [81, 7, 74]. Following are the relevant concepts and theorems of Lipton and Young [74].

Definition 2.3.1 (Zero-Sum Game) *A (finite)* zero-sum game *between two players, here called a* row player *and a* column player*, is represented as a real $m \times n$ matrix M. If the row player plays strategy i and the column player plays strategy j, then the row player receives payoff $M(i,j)$, and the column player receives payoff $-M(i,j)$.*

Definition 2.3.2 (k-Uniform Distribution) *Let S be any set, and let $k \in \mathbb{N}$. A* k-uniform distribution on S *is a probability distribution obtained by choosing uniformly from a multiset of k elements from S.*

Definition 2.3.3 (Circuit-Input Game) *Let $\mathcal{C}$ be a set of n circuits, let $\mathcal{I}$ be a set of m inputs, and let M be an $m \times n$ matrix with entries in $[0,1]$. (Intuitively, $M(C,x)$ represents some cost of the computation of $C(x)$.) The* circuit-input game w.r.t. *M is the two-player zero-sum game given by the matrix M.*

For a function $f : \{0,1\}^\star \to \{0,1\}$, the central game of interest is the *circuit-input game for f on size-s circuits and n inputs*, which is the circuit-input game w.r.t. the following matrix M. M has 2^n rows (for all strings x in $\{0,1\}^n$) and $2^{O(s \log s)}$ columns (for all circuits C of size s), and

$$M(C,x) := \begin{cases} 0 & \text{if } C(x) = f(x) \\ 1 & \text{otherwise} \end{cases}$$

Let $\mathcal{C}$, $\mathcal{I}$, and M be as above. Assume without loss of generality that $\exists C_0, x_0$ such that $M(C_0,x_0) = 0$ and $\exists C_1, x_1$ such that $M(C_1,x_1) = 1$. We shall use the following two theorems saying that approximately optimal strategies with small support exist for every game M:

Theorem 2.3.1 ([81, 7, 74]) *Let $\epsilon > 0$, let $k > \dfrac{\ln|\mathcal{I}|}{2\epsilon^2}$, and let $\ell > \dfrac{\ln|\mathcal{C}|}{2\epsilon^2}$.*

1. *There exists a k-uniform distribution p on $\mathcal{C}$ such that for every $x \in \mathcal{I}$, the expectation $\mathop{\mathbf{E}}_{C \sim p}[M_{C,x}] < \mathcal{V}(M) + \epsilon$, where $\mathcal{V}(M)$ denotes the value of the circuit-input game w.r.t. M.*

2. *There exists an ℓ-uniform distribution p on $\mathcal{I}$ such that for every $C \in \mathcal{C}$, the expectation $\mathop{\mathbf{E}}_{x\sim p}[M_{C,x}] > \mathcal{V}(M) - \epsilon$, where $\mathcal{V}(M)$ denotes the value of the circuit-input game w.r.t. M.*

This theorem can be proved using a random sampling argument and standard large deviation (Chernoff-Hoeffding) bounds. From Theorem 2.3.1, we may derive the following general consequence which does not appear in prior work:

Theorem 2.3.2 *Let $\mathcal{C}_n$ be a set of 2^t circuits where each circuit has n inputs, let $\mathcal{I}_n \subseteq \{0,1\}^n$, and let $f : \{0,1\}^n \to \{0,1\}$. Let $\epsilon : \mathbb{N} \to \mathbb{Q} \cap (0,1]$, and let $p, q : \mathbb{N} \to [0,1]$ with $p + q \leq 1 - \epsilon$. For every $n \in \mathbb{N}$, one of the following must hold:*

1. *There exists an $O(n/\varepsilon(n)^2)$-size multiset $X_n \subseteq \mathcal{C}_n$ such that for every $y \in \mathcal{I}_n$, $C(y) = f(y)$ for more than a $p(n)$ fraction of the circuits $C \in X_n$.*

2. *There exists an $O(t/\varepsilon(n)^2)$-size multiset $Y_n \subseteq \mathcal{I}_n$ such that for every $C \in \mathcal{C}_n$, $C(y) \neq f(y)$ for more than a $q(n)$ fraction of the inputs $y \in Y_n$.*

Proof. Let M be the circuit-input game for a function f. Let ε, p, q, and n be as in the statement of the theorem. Set a parameter $\delta := \varepsilon(n)/2$, set $k := \dfrac{\ln|\mathcal{I}_n|}{\delta^2} = \dfrac{O(n)}{\delta^2}$, and set $\ell := \dfrac{\ln|\mathcal{C}_n|}{\delta^2} = \dfrac{O(t)}{\delta^2}$. Then by Theorem 2.3.1, there exists a k-uniform distribution X_n on $\mathcal{C}_n$ such that for all $y \in \mathcal{I}_n$,

$$\mathop{\mathbf{E}}_{C\sim X_n}[M[C,y]] < \mathcal{V}(M) + \delta, \tag{2.1}$$

and there exists an ℓ-uniform distribution Y_n on $\mathcal{I}_n$ such that for every $C \in \mathcal{C}_n$,

$$\mathop{\mathbf{E}}_{y\sim Y_n}[M[C,y]] > \mathcal{V}(M) - \delta. \tag{2.2}$$

Assume that there exists $y^* \in \mathcal{I}_n$ such that $C(y^*) \neq f(y^*)$ for at least a $1 - p(n)$ fraction of the circuits $C \in X_n$. Since M is a Boolean matrix, we have for every

$C^* \in \mathcal{C}_n$ that

$$\begin{aligned} q(n) &\leq 1 - p(n) - \varepsilon(n) \\ &\leq \mathop{\mathbf{Pr}}_{C \sim X_n}[C(y^*) \neq f(y^*)] - \varepsilon(n) \quad \text{(by choice of } y^*\text{)} \\ &< \mathcal{V}(M) - \delta \quad \text{(by choice of } \delta \text{ and (2.1))} \\ &< \mathop{\mathbf{Pr}}_{y \sim Y_n}[C^*(y) \neq f(y)]. \quad \text{(by (2.2))} \end{aligned}$$

This completes the proof. □

That is, we can trade off between the "measure of success" p of the succinct strategy for the circuit player, and the measure of success q of the succinct strategy for the row player. This tradeoff can be exploited for complexity-theoretic purposes as follows: if item 1 does not hold (because of circuit lower bounds for computing f with $\mathcal{C}$ circuits) then there are small multisets "witnessing" this inability to compute. Lipton and Young observed this consequence for the special case of $p = 1/2$ and $q = 1/2 - \varepsilon$, calling such sets *anti-checkers*. In our main results, we shall adjust p and q to different values, as needed. (For example, in our results concerning natural properties, we use the case of $p = 0$ and $q = 1 - \varepsilon$.)

The problem of approximately solving a zero-sum game has been studied in operations research as well; for example, Grigoriadis and Khachiyan [54] show how to find an approximately optimal strategy with randomness in time sublinear in the size of the game matrix.

Bshouty *et al.* [29] studied the circuit-input game in the context of learning theory, focusing on the complexity of finding succinct strategies in the game. They proved (for example) that if $\mathsf{NP} \subset \mathsf{P/poly}$ then one can uniformly construct circuits solving SAT in $\mathsf{ZPP}^{\mathsf{NP}}$ – this gave a new collapse of the polynomial-hierarchy under the assumption that $\mathsf{NP} \subset \mathsf{P/poly}$. Dually, if $\mathsf{NP} \not\subset \mathsf{P/poly}$, then their results also show that one can uniformly construct multisets of satisfiable formulas that "fool" all small circuits, in $\mathsf{ZPP}^{\mathsf{NP}}$: Fortnow, Pavan, and Sengupta applied this consequence to the "two queries" problem in structural complexity [46]. Subsequently, Fortnow *et al.* [45] proved that the problem of finding approximate succinct strategies in an implicitly

represented game (such as the circuit-input game) is promise-S_2-complete. Other works on succinct games include [30, 95].

When circuit lower bounds are true, succinct strategy results like Theorem 2.3.2 tell us that there are small distributions of inputs that "fool" all small circuits. Another class of related results have focused on a different flavor of hardness result: *given the code* of an efficient algorithm (randomized or otherwise) that's supposed to solve SAT, one can construct even smaller distributions that fool the given algorithm [56, 12, 25].

It is natural to ask whether Theorem 2.3.2 can be improved, so that $p + q = 1$. We now show that this is not possible, at least not in full generality. In particular, while Theorem 2.3.2 holds for all $p + q \leq 1 - \varepsilon$, where $\varepsilon > 0$, it does not hold for $p = q = 1/2$ and all matrices M:

Theorem 2.3.3 *Theorem 2.3.2 does not hold with $p = q = 1/2$.*

Proof. Let $f : \{0,1\}^* \to \{0,1\}$ be a Boolean function without circuits of size $c \cdot ns$, for a sufficiently large constant $c \geq 1$. Suppose Theorem 2.3.2 holds with $p = q = 1/2$ and the circuit-input game for f on size-s circuits and n inputs. Then at least one of the following holds.

1. There is an $O(n)$-size multiset $\mathcal{C}_n$ of size-s circuits where $\Pr_{C \in \mathcal{C}_n}[C(x) \neq f(x)] < 1/2$ holds for all $x \in \{0,1\}^n$.

2. There exists an $O(s \log s)$-size multiset $\mathcal{X}_n$ of n-bit inputs such that for every circuit C of size s, $\Pr_{x \in \mathcal{X}_n}[C(x) \neq f(x)] \geq 1/2$.

In the first case, f can be implemented with a (strict) MAJORITY circuit on $\mathcal{C}_n$, giving a circuit of size $O(ns)$, contradicting our choice of f.

The second case also leads to a contradiction. For any $\mathcal{X}_n$, we may take a circuit C of size $O(n)$ that has a single input x in $\mathcal{X}_n$ hardwired along with the value $f(x)$. The circuit C outputs $f(x)$ on input x, and otherwise it outputs the bit b maximizing the quantity $\Pr_{x' \in \mathcal{X}_n \setminus \{x\}}[f(x') = b]$. It follows that $\Pr[C(x) \neq f(x)] < \frac{1}{2}$. □

Chapter 3

A Potential Circumvention of Natural Proofs

3.1 Natural Proofs

In this chapter, we apply Theorem 2.3.2 to circumvent the proof barrier of Razborov and Rudich [91]. First, we present the necessary background and definitions. In the following, let Γ and $\mathcal{C}$ be complexity classes.

Definition 3.1.1 (Property of Boolean Functions) *A* property *of Boolean functions is a set of truth tables.*

For the purposes of this chapter, there are several kinds of properties that we care about in particular.

Definition 3.1.2 (Constructivity) *A property P of Boolean functions is* Γ-constructive *if $P \in \Gamma$.*

The constructivity condition simply states that P can be computed efficiently, in the sense that there is some efficient (as defined by Γ) algorithm A that, given the truth table of a Boolean function f, determines whether $f \in P$.

Definition 3.1.3 (Largeness) *A property P of Boolean functions is* large *if for every $n \in \mathbb{N}$, $\left|P \cap \{0,1\}^{2^n}\right| \in 2^{2^n - O(n)}$.*

Note that for a natural number n, there are 2^n n-bit strings, so there are 2^{2^n} Boolean functions on n bits. Largeness states that P contains a $2^{-O(n)}$ fraction of these functions. Each is represented as a 2^n-bit truth table, so $2^{-O(n)}$ actually means inverse polynomial in the length of the truth tables in question. In other words, the largeness condition essentially states that a property contains a non-negligible fraction of all Boolean functions.

Definition 3.1.4 (Usefulness) *A property P of Boolean functions is* useful against $\mathcal{C}$ *if for infinitely many $n \in \mathbb{N}$, $P \cap \mathcal{C} \cap \{0,1\}^{2^n} = \emptyset$.*

The usefulness condition states that (infinitely often) P contains no functions from (i.e. computable in) $\mathcal{C}$. Note that the complexity class $\mathcal{C}$ in the usefulness condition constrains the complexity of the functions satisfying the property P. In contrast, Γ from the constructivity condition constrains the complexity of P itself.

Definition 3.1.5 (Natural Property) *A* Γ-natural property useful against $\mathcal{C}$ *is a property P that is Γ-constructive, large, and useful against $\mathcal{C}$.*

Suppose a function f satisfies a Γ-natural property useful against $\mathcal{C}$. Because of the usefulness condition, this constitutes a proof that $f \notin \mathcal{C}$, hence the term *natural proof*. However, Razborov and Rudich observed that because of the constructivity and largeness conditions, this also gives a Γ algorithm that can distinguish any pseudorandom generator candidate computable in $\mathcal{C}$ from a truly random string with non-negligible probability, which essentially means that all cryptography implemented in $\mathcal{C}$ is insecure against Γ. Since it is widely believed that there are pseudorandom generators implementable in P/poly and secure against P, this means that there should not be any P-natural properties useful against P/poly. In particular, no such natural property should be able to separate NP from P/poly.

3.2 Natural Properties from Circuit Lower Bounds for SAT

If we consider only self-checking circuits, the above proof barrier falls apart. In this chapter, we prove that natural properties useful against self-checking circuits are *equivalent* to circuit lower bounds in some important settings. To see this, consider NP vs P/poly and the SAT problem. Take any circuit C (which purportedly solves SAT). We convert C into a new circuit C' which never errs when it reports satisfiability. C' contains several copies of C, and when C reports satisfiability, C' checks it by trying to generate a satisfying assignment. C' iteratively plugs in values for a variable of the supposedly satisfiable formula and queries C on the reduced formulas to see which (if any) is still satisfiable. One of two things can occur. In one case, C will eventually contradict itself by saying that a formula is satisfiable, but both of the reduced formulas are unsatisfiable (or by saying that the formula FALSE is satisfiable). In this case, C' aborts and reports "UNSAT". Otherwise, C' will eventually exhaust all of the variables of the original formula, at which point it has generated a satisfying assignment and can report "SAT" without error. Hence we may assume without loss of generality that a circuit never errs when it reports satisfiability.

Let SAT_n denote the restriction of SAT to formulas encoded in n bits.

Definition 3.2.1 (SAT Solver) *A Boolean circuit C on n inputs is a* SAT solver *if its truth table is pointwise dominated by SAT_n.*

That is, for all n-bit formulas φ, $C(\varphi) = 1$ implies that φ is satisfiable. (Such circuits are called "SAT solvers" because one could use these circuits to print satisfying assignments to formulas, when the circuits report "SAT.") The class of functions computable by polynomial-size SAT solvers is an expressive class, including special cases such as 2-SAT, Horn-SAT, etc. By the previous paragraph, to prove $\mathsf{NP} \not\subset \mathsf{P/poly}$ it suffices to prove that no polynomial-size family of SAT solvers can compute SAT. That is, a natural proof need only be useful against small SAT solvers, rather than all small circuits. It turns out that such natural proofs must exist, assuming

$\mathsf{NP} \not\subset \mathsf{P}/\text{poly}$.

Theorem 3.2.1 (recall Theorem 1.3.1) *$\mathsf{NP} \not\subset \mathsf{P}/poly$ if and only if there is an $\mathsf{AC}^0/n^{o(1)}$-natural property[1] that is useful against polynomial-size SAT solvers and accepts SAT_n for all n.*

Proof. One direction of the equivalence is trivial: if there is any logical property that is false on the truth tables of all polynomial-size SAT_n solvers for infinitely many n, yet the property is true of SAT_n for all n, then no polynomial-size SAT solving circuit can compute SAT_n almost everywhere. Therefore $\mathsf{NP} \not\subset \mathsf{P}/\text{poly}$.

Now we proceed with the other direction. Assume $\mathsf{NP} \not\subset \mathsf{P}/\text{poly}$. Let s be a polynomial in n. For every $n \in \mathbb{N}$, let $\mathcal{C}_n$ denote the set of n-input circuits of size $s(n)$ which are SAT solvers (i.e., they never err on unsatisfiable formulas). Set $\mathcal{I}_n := \text{SAT}_n$, i.e., the set of satisfiable formulas encoded in n bits. Consider the circuit-input game M for SAT, over the set of circuits $\mathcal{C}_n$ and set of inputs $\mathcal{I}_n$.

Applying Theorem 2.3.2 to this game M (taking $p(n) = 0$ and $q(n) = 1 - \varepsilon(n)$ for some inverse polynomial $\varepsilon(n)$), either:

1. there is an $\text{poly}(s, n)$-size set $X \subseteq \mathcal{C}_n$ such that for every $x \in \mathcal{I}$, at least one $C \in X$ computes $\text{SAT}(x)$, or

2. there is a $\text{poly}(s, n)$-size set $Y \subseteq \mathcal{I}_n$ such that every circuit $C \in \mathcal{C}_n$ computes SAT correctly on at most an $\epsilon(n)$ fraction of inputs in Y.

In the first case, we may construct a polynomial-size circuit for SAT_n by simply taking the OR of the circuits in X: since the circuits never err on unsatisfiable formulas, this will compute SAT_n. Our assumption $\mathsf{NP} \not\subset \mathsf{P}/\text{poly}$ is therefore contradicted if the first case holds for almost every n.

Suppose the second case holds for infinitely many n. Then we can construct an algorithm A which takes as input the 2^n-bit truth table of a function $f : \{0,1\}^n \to$

[1]Recall Definitions 2.2.15 and 2.2.11. The complexity class $\mathsf{AC}^0/n^{o(1)}$ is the set of problems that can be solved with uniformly generated circuit families that consist of AND and OR gates, have constant depth and polynomial size, and take advice strings of length $n^{o(1)}$.

$\{0,1\}$ and is armed with Y as advice. Algorithm A accepts f if f outputs 1 on at least a $2\varepsilon(n)$ fraction of the inputs in Y, and rejects f if f outputs 1 on at most an $\varepsilon(n)$ fraction of the inputs in Y. This A is implementable in polynomial-size AC^0 (in fact, depth-3 $\textsc{And} \circ \textsc{Or} \circ \textsc{And}$ circuits), by classical results on distinguishing strings with many 1's from strings with many 0's [2, 102]. Furthermore, A trivially accepts SAT_n for every n, because $\mathrm{SAT}_n(y) = 1$ for every input $y \in Y$.

Notice that, while A rejects the truth tables of SAT solving circuits of $s(n)$ size, A will *accept* a randomly chosen Boolean function with probability $1 - o(1)$. Notice that the advice needed is $s(n) \leq O(n^k)$, which is polylogarithmic in the input length, 2^n. Therefore, A is an $\mathsf{AC}^0/n^{o(1)}$-natural property that is useful against SAT solving circuits of $s(n)$ size. Such an A can be constructed for every polynomial $s(n)$, assuming $\mathsf{NP} \not\subset \mathsf{P/poly}$. □

It is worth noting that the above theorem holds not only for SAT solvers, but for any circuit which computes a NP-complete problem with one-sided error, regardless of whether the problem in question exhibits self-reducibility. We consider SAT specifically because self-reducibility allows us to construct a SAT solver from an arbitrary circuit with only polynomial overhead.

As it turns out, the natural property in Theorem 3.2.1 can be made uniform. However, the proof is not particularly illustrative.

Theorem 3.2.2 (cf. Theorems 1.3.1, 3.2.1) $\mathsf{NP} \not\subset \mathsf{P/poly}$ *if and only if there is a uniform* AC^0*-natural property that is useful against polynomial-size SAT solvers and accepts* SAT_n *for all* n*.*

Proof. Again, one direction is trivial. For the other direction, suppose $\mathsf{NP} \not\subset \mathsf{P/poly}$. Consider the algorithm A that accepts a function f if and only if f is not *strictly* dominated by SAT_n.

We check the three conditions of a natural property:

- Constructivity: SAT_n can be computed uniformly with a circuit of size polynomial in the input length 2^n, by exhaustively checking all possible assignments

(at most 2^n) on all (2^n) possible n-bit formulas. Each check is a CNF evaluation, which can be computed in uniform AC^0. The checks can be combined into a truth table for SAT_n using a layer of 2^n OR gates. The resulting truth table can be compared to the input truth table using a 2-DNF. Hence A is uniform AC^0-constructive.

- Largeness: There is at least one unsatisfiable formula φ. A accepts all functions that accept φ, i.e. at least half of all Boolean functions.

- Usefulness: A accepts SAT_n itself but no other functions computable by SAT solvers (of any size). SAT_n cannot be computed by a polynomial-size SAT solver by assumption, so A is useful against polynomial-size SAT solvers.

□

3.3 Natural Properties from Circuit Lower Bounds for Checkable Functions

It is easy to extend Theorem 3.2.1 to general functions f with deterministic one-sided checkers: functions f that allow polynomial-size circuits with oracle gates for f which never err when they output 0 (or never err when they output 1). Now we consider languages which have (randomized) polynomial time program checkers, such as EXP-complete languages [13, 24], PERMANENT [72], etc. Such randomized program checkers can be adapted to give polynomial size circuit families which act as program checkers that *deterministically* check all functions computable by small circuits. These deterministic program checkers can then be used to prove (as above with SAT) equivalences between circuit lower bounds and the existence of natural properties. Here we use the following definition of a program checker.

Definition 3.3.1 (Program Checker) *Let $L \in \{0,1\}^*$. A* randomized (polynomial time) program checker for *L is a randomized (polynomial time) algorithm with*

oracle access to an arbitrary function f, which when given an n-bit input x and randomness r will accept with probability greater than $2/3$ (over the choice of r) if $L = f$ and will reject with probability greater than $2/3$ if $L(x) \neq f(x)$.

In order for a randomized program checker to be adapted to produce a deterministic circuit family, we require that the language L be paddable, so that a circuit computing L on inputs of length n can also compute L on inputs of length less than n. Hence for the rest of the section, we consider only functions which are paddable in the following sense.

Definition 3.3.2 (Paddability) *A language L is* paddable *if there exists a language L' such that $L = \left\{x01^k : x \in L', k \in \mathbb{N}\right\}$.*

Note that any language L' can be converted into a paddable language L as above, while preserving both asymptotic circuit complexity (up to a polynomial factor) and the existence of program checkers.

Theorem 3.3.1 *Let p and t be polynomials, and let L be a paddable language with a randomized $t(n) - n$ time program checker. Then there exists a polynomial size circuit family which deterministically checks all functions on n inputs computable by circuits of size at most $p(t(n))$.*

Proof. Let p and t be polynomials in n. Let L be any paddable language with a randomized $t(n)-n$ time program checker, i.e. there exists a randomized $t(n)-n$ time algorithm A such that for every input $x \in \{0,1\}^n$ and every $f : \{0,1\}^{t(n)} \to \{0,1\}$, $A^f(x) = 1$ with probability more than $2/3$ if f is the $t(n)$th slice of L, and $A^f(x) = 0$ with probability more than $2/3$ if $f\left(x1^{t(n)-n}\right) \neq L(x)$.

We may amplify this success probability to $1 - 2^{-q(n)}$ (for some polynomial $q(n) > 2p(t(n)) \log p(t(n)) + n$), by creating another checker A' which runs A independently $18q(n)$ times and returns the MAJ of the $18q(n)$ results; appealing to a Chernoff bound yields the higher success probability.

We may simulate A' with a polynomial size family $\{B_n\}$ of oracle circuits which take as input the string $x \in \{0,1\}^n$ and a string of randomness $r \in \{0,1\}^{n^k}$, and which use oracle gates to compute the function f. Since there are at most $2^{2p(t(n))\log p(t(n))}$ circuits of size $p(t(n))$ and 2^n n-bit input strings, we have (using a union bound) with non-zero probability (over the choice of $r \in \{0,1\}^{n^k}$), for every $x \in \{0,1\}^n$ and every f computable by a circuit of size $p(t(n))$,

- $f(z) = L(z)$ for all $z \in \{0,1\}^{t(n)} \implies B_n^f(x,r) = 1$ and
- $f(x) \neq L(x) \implies B_n^f(x,r) = 0$.

Hence there is some choice of randomness $r^* \in \{0,1\}^{n^k}$ for which $B_n(-,r^*)$ is a deterministic program checker (where the randomness r^* is hard coded into the circuit). □

Fix a polynomial p and a circuit family A_n which deterministically checks circuits of size at most $p(n)$. Now for any circuit $\{C\}$ with $t(n)$ inputs and of size $p(t(n))$, we may augment C with $A_{t(n)}$ to create a self-checking circuit C' on n inputs with polynomial overhead. We may treat C' as a ternary circuit which outputs 0 (resp. 1) if $A_{t(n)}$ outputs 1 and C outputs 0 (resp. 1), and which outputs $\perp$ if $A_{t(n)}$ outputs 0. We may treat $\perp$ as 0 or 1, in which case C' gives a circuit with one-sided error (in either direction). Call all such C' the *self-checked circuits for L*.

Theorem 3.3.2 *Let L be a paddable language with a randomized polynomial time program checker. If $L \notin \mathsf{P}/poly$, then for every polynomial s, there exists a natural property computable in $\mathsf{AC}^0/n^{o(1)}$ useful against size-$s(n)$ self-checked circuits for L.*

Proof. In the case where $\perp = 0$ above, we may take $\mathcal{C}_n$ to be the set of $s(n)$ size circuits, $\mathcal{I}_n := \{0,1\}^n \cap L$, and $M(C,y) := 1 - C(y)$. From Theorem 2.3.2 (taking $p(n) = 0$ and $q(n) = 1 - \epsilon(n)$ for some inverse polynomial ϵ), either there is a polynomial size set $X \subseteq \mathcal{C}_n$ such that for every $y \in \mathcal{I}$, at least one $C \in X$ computes $L(y)$, or there is a polynomial size set $Y \subseteq \mathcal{I}_n$ such that every circuit $C \in \mathcal{C}_n$ computes L correctly on at most an $\epsilon(n)$ fraction of inputs in Y. If the

former case holds for almost every n, then we may construct a polynomial size circuit family for L by taking the OR of the circuits in X. Otherwise, the latter case holds infinitely often, giving an efficiently computable ($\mathsf{AC}^0/n^{o(1)}$) natural property useful against functions with one-sided error which are computable with $s(n)$ size circuits.

Note that the choice to treat $\perp$ as 0 is arbitrary. We may instead treat $\perp$ as 1, in which case Theorem 2.3.2 either gives a polynomial size set of circuits whose AND computes L, or an $\mathsf{AC}^0/n^{o(1)}$-computable natural property useful against functions with $p(n)$ size circuits and which are (bitwise) at least L. □

3.4 New Lower Bounds?

These results raise questions regarding the circumvention of natural proofs that seem worthwhile to explore further. For instance, can new circuit lower bounds be proved, based on the guidance of Theorem 3.2.1? Again, this theorem tells us that we should expect there to be combinatorial properties useful against polynomial-size SAT solving circuits, or in general, circuits which never err when they print solutions to their input instances. To be more concrete, let $\text{CLIQUE}_{n^2}^{n/2} : \{0,1\}^{n^2} \to \{0,1\}^n$ be the function which treats its input as an $n \times n$ adjacency matrix A, and outputs a bit vector specifying a clique of size at least $n/2$ in A, when one exists (otherwise, it outputs the all-zeros vector). Can one prove that computing $\text{CLIQUE}_{n^2}^{n/2}$ requires circuits of size at least $4n^2$ over the basis B_2?

Chapter 4

Circuit Lower Bounds as Data Design Problems

4.1 Data Complexity

In this chapter, we use succinct strategies for zero-sum games to set up a framework that is "dual" to the usual computational view of circuits computing functions on inputs, treating inputs as the "programs" and circuits as the "input data".

Consider a language L, called the *primal* language. The primal complexity measure is the standard measure of the circuit complexity of L, as a function of the number of input bits.

We now define the *primal* language:

Definition 4.1.1 (Slice of a Boolean Function) *The* n^{th} slice *of* L *is* $L_n := L \cap \{0,1\}^n$.

Definition 4.1.2 (Dual Language) *The* dual language TEST-L *is the set of descriptions of circuits that are consistent with* L*, in the sense that they compute a slice of* L.

However, TEST-L is not computed in the usual sense, with a circuit. Instead, we input strings as a model of computation. TEST-L can be computed with a test suite

of inputs by checking to see that a given circuit computes L correctly on all inputs in the test suite. Say that such a test suite computes TEST-L correctly if every circuit that does not compute a slice of L errs on at least one input in the test suite.

We can now define a dual complexity measure.

Definition 4.1.3 (Data Complexity) *The* data complexity *of L is the smallest number of inputs required by a test suite to compute* TEST-L*, as a function of the size of the circuit being tested.*

Note that the data complexity is a function of the size of the *circuit* being tested, and not the size of the input that this circuit takes. This is because the circuit itself is the input to the problem TEST-L. Every language L can be computed by trivial circuits of exponential size. However, the theory of circuits becomes interesting when only small circuits are considered. Similarly, there are also always trivial test suites of exponential size that can conclusively determine whether a circuit correctly computes L. It would suffice to try all possible inputs of size up to the size of the circuit being tested. However, a natural goal is to design smaller test suites that still compute TEST-L correctly.

Our ideas for using data to test circuits for a function are vaguely related to the notion of *teaching dimension of a class of concepts*, from learning theory (see Goldman and Kearns [50] and Shinohara and Miyano [96]). The teaching dimension is defined with respect to a *collection* of functions, and it bounds the total number of labeled examples needed to identify each concept in the collection (to distinguish it from the other concepts). However, this notion is information-theoretic: there are no computational bounds placed on *what* is distinguishing one function from another. In our setting, we have a collection of *programs* and a specific function f of interest, and wish to know how many labeled examples we need to distinguish "bad" programs which do not compute f from "good" ones which do.

Mulmuley's GCT program [77] has also considered "sets of counterexamples" similar to our test suites, calling them "obstructions."

4.2 Duality with Circuit Complexity

The remainder of this chapter proves that the above definitions do indeed present a reasonable duality. For simplicity, we prove the main duality theorem (Theorem 4.2.1) only for the class of circuits over the basis B_2 of all two-bit Boolean functions, although analogous statements will hold for any complete basis with minor modifications.

The data complexity of testing size-s circuits is always at most $2^{O(s)}$ for any language L: one can simply include all possible input/output pairs on inputs of length up to s. We are interested to know: for what languages L can the data complexity be much smaller? We prove an equivalence between upper bounds on the data complexity of L and lower bounds on the circuit complexity of L:

Theorem 4.2.1 *Let $L \subseteq \{0,1\}^\star$, and let $S(n) \geq 2n$ for all n.*

1. *If $L \in \textsc{Size}(S(n))$, then the data complexity of L is at least $2^{\Omega(S^{-1}(s))}$ almost everywhere.*

2. *If L is not in $\textsc{Size}(n \cdot S(n))$, then the data complexity of L is at most $O(2^{S^{-1}(s)} + S^{-1}(s) \cdot s^2 \log s)$ infinitely often.*

Since for a uniformly random language $L_n \subseteq \{0,1\}^n$, $L_n \notin \textsc{Size}(2^n/n^2)$ with high probability, we have the following corollary:

Corollary 4.2.1 *If $L \subseteq \{0,1\}^*$ is uniformly random, then almost certainly the data complexity of L is at most $O(s^2 \log^2 s)$ infinitely often.*

We also note that since the circuit complexities of a language L over any complete bases differ by at most a constant factor (and as noted previously, all results presented hold for an arbitrary complete gate basis with minor modifications to the proofs), the data complexities of L over any complete bases differ by at most a constant factor.

To get some intuition towards a proof of Theorem 4.2.1, notice that if we replace "data complexity" with "time complexity" in the above, one direction of the equivalence is easy to establish. Namely, for languages L computable within exponential

time, when the circuit complexity of L is large, the *time complexity* of TEST-L will be provably low, as follows.

Suppose $S(n)$ is a lower bound on the circuit complexity of computing L on n-bit inputs. To efficiently test a given circuit C of size s with n inputs, we can immediately reject if $s < S(n)$, otherwise we may try all $2^n < 2^{S^{-1}(s)}$ inputs to C and check whether the truth table obtained for C matches L_n on n-bit inputs. For L computable within $2^{O(n)}$ time, this algorithm takes $2^{O(S^{-1}(s))}$ time; larger $S(n)$ entails a faster running time. (For example, if some L in $2^{O(n)}$ time requires $2^{\varepsilon n}$ size $\mathcal{C}$-circuits, then testing $\mathcal{C}$ for f is in poly(s) time.) However, this particular connection is not terribly useful: we are basically saying that strong circuit lower bounds happen to make testing circuits for L trivial, because most circuits can be immediately rejected. Moreover we do not know if low time complexity for testing circuits for L will imply analogous circuit lower bounds for L, in general.

The equivalence between circuit complexity and data complexity is far less obvious. We use the following consequence of results on the circuit-input game (Theorem 2.3.2): when circuits are too small to compute a function, there are small data sets that will efficiently refute these small circuits.

Proof of Theorem 4.2.1. (Part 1.) Suppose for all n, there is a circuit C of size $S(n)$ which computes L_n on all inputs of length n. For each n, we claim that every test set T_s for L on circuits of size $S'(n) = S(n) + n$ satisfies $|T_{s'}| \geq 2^n$. Observe that for every circuit C of size $S(n)$ with n inputs and for all x of length n, there is a circuit C_x of size at most $S'(n) = S(n) + n$ which agrees with C on all inputs except x, where C_x and C disagree. In particular, C_x can use a tree of $n - 1$ gates that outputs 1 if and only if the input equals x, and take the XOR of this tree's output with the circuit C.

So given an n-input circuit C computing L with size at most $S'(n)$, in order to distinguish C from all of the C_x, x must be included in $T_{s'}$. That is, all x of length $n = \Omega(S^{-1}(s))$ must be in $T_{s'}$.

(Part 2.) Suppose the circuit complexity of L is greater than $n \cdot S(n)$ on inputs of

length n. Then there cannot be a collection $\mathcal{D}$ of $O(n/\varepsilon^2)$ size-$S(n)$ circuits such that for all $x \in \{0,1\}^n$, $C(x) = L(x)$ for more than a $1/2$ fraction of C in $\mathcal{D}$: otherwise, taking the MAJ of the outputs of circuits in $\mathcal{D}$ would yield a circuit for L of complexity at most $n \cdot S(n)$. Therefore, item 1 of Theorem 2.3.2 does not hold with $p = 1/2$, and hence item 2 must hold for $q \leq 1/2 - \varepsilon$ for every sufficiently small ε. Setting ε appropriately, this implies for all input lengths m ranging from n to $n \cdot S(n)$, there is a set $X_{m,s}$ of m-bit strings x with cardinality $O(S(n) \log S(n))$, such that every circuit of size $S(n)$ taking m bits of input fails to compute L correctly on at least $1/10$ of the x in the set $X_{m,s}$. By adding all strings in $X_{m,s}$ to the set X_s, we can refute any circuit with more than n inputs and size $S(n)$, with a set of $O(nS(n)^2 \log S(n))$ strings.

For input lengths m that are below n, there may be a size $S(n)$ circuit for L that works on all m-bit strings. To cover this case, we simply include all bit strings of length up to $n-1$ in the set X_s as well – then, every circuit of size s and at most n inputs can also be checked. In total, we have a test set X_s of cardinality $O(2^n + nS(n)^2 \log S(n))$. For size s circuits, we set $s := S(n)$, i.e., $n = S^{-1}(s)$, so the cardinality is $O(2^{S^{-1}(s)} + S^{-1}(s) \cdot s^2 \log s)$ as a function of the circuit size s. □

In the above theorems, the small cardinality test suites X_s have the following structure: for input sizes which are "too long" to support size-s circuits for L, we have small sets of counterexamples from the circuit-input game, but as the input sizes decrease, we reach a threshold where it's possible to compute L within size s, and must start including all possible inputs.

When we consider polynomial-size circuits in general, we simply obtain an *equivalence*:

Corollary 4.2.2 *A language L is in P/poly if and only if for some $\varepsilon > 0$, the data complexity of L is greater than 2^{s^ε} for almost every s.*

Proof. If $L \in \mathsf{P}/\mathrm{poly}$, then it is in $\textsc{Size}(n^k)$ for some constant k. By Part 1 of Theorem 4.2.1, the data complexity of L is at least $2^{\Omega(s^{1/k})}$. On the other hand, if $L \notin \mathsf{P}/\mathrm{poly}$, then it is not in $\textsc{Size}(n^{k+1})$ for every k. By Part 2 of Theorem 4.2.1, the

data complexity of L is then at most $O(s^k \log s + 2^{s^{1/k}})$ for all k. $\square$

Corollary 4.2.3 (recall Theorem 1.4.1) $\mathsf{NP} \not\subset \mathsf{P}/poly$ *(resp.* $\mathsf{NP} \not\subset \mathsf{i.o.P}/poly$*) if and only if for every* $\varepsilon > 0$ *and for infinitely many* s *(resp. for every* $\varepsilon > 0$ *and for every* s*), the data complexity of SAT is at most* $O\left(2^{s^\varepsilon}\right)$.

4.3 Open Questions

We conclude the chapter with two open questions of particular interest to the authors.

1. *Can the equivalence of Theorem 4.2.1 be tightened further?* Currently there is a gap between the two implications in the equivalence, amounting to a multiplicative factor of n. Could this gap be necessary?

2. *How does the complexity of L relate to the complexity of* $\textsc{Test} - L$*?* Here we mean "complexity" in the usual, most generic sense: if L is known to be computable in some particular complexity class, what can we say about the complexity class(es) that support testing for L?

Chapter 5

Black-Box Hypotheses

5.1 A Complexity Theoretic View of Obfuscation

What kind of code "behaves" like a black box to any code analyst? In particular, what programs are so difficult to analyze that every potential analyst can discern essentially no information from the code, other than its input-output behavior? Such questions are of great importance in cryptography and formal verification: what sort of code is difficult to verify without considerable resources? What kind of code can be obfuscated? What properties of functions can be automatically tested?

A priori, the answers to such questions depend on three factors:

1. The complexity of the code: what instructions are allowed in the code, the computational complexity (e.g. time/space/size/depth complexity) of the algorithm implemented by the code, and so on.

2. The complexity of the analyst: what sorts of operations the analyst can perform, and how much resources it has (time/space/size/depth) to analyze the code.

3. The actual function being computed by the code. If the function itself is trivial or extremely complicated, this could affect how "black box" it can possibly look.

In the pioneering work of Barak *et al.* [14] on obfuscation, the authors proposed a compelling conjecture about black-box obfuscation that they called a "Scaled-Down

Rice's Theorem" [14, Conjecture 5.1]; the conjecture has recently been renamed the *Black Box Hypothesis* (BBH) [93, 65]. Informally, the Black-Box Hypothesis posits that, when code is represented as a small Boolean circuit, and a code analyst is represented as an efficient algorithm, the only possible analysis tasks are ones that could have been performed using only the input-output behavior of the code (and not the code itself).

The original Black-Box Hypothesis is still a major open problem, but other natural variants of the hypothesis may be more tractable to resolve *unconditionally*. Here we consider variants of the Black-Box Hypothesis in a more general complexity-theoretic setting, where the complexity of the analyst, the complexity of the code being analyzed, and the function to be obfuscated (the "box") are taken carefully into account. For example, of particular interest are the cases where the "analyst function" is taken from a "low" complexity class $\mathcal{A}$ (smaller than P), and the box is also from a "low" complexity class $\mathcal{C}$.

More formally, we study abstract forms of the Black-Box Hypothesis (sometimes abbreviated as BBH in the following). Let $\mathcal{C}$ be a set of circuits and let $\mathcal{A}$ be a complexity class that permits oracles in its definition. We say that a property $P : \mathcal{C} \to \{0,1\}$ of $\mathcal{C}$ is *semantic* if $P(C) = P(C')$ for all pairs of circuits C and C' in $\mathcal{C}$ which compute the same function.

Hypothesis 5.1.1 ($\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$) [Informal Statement, cf. Hypothesis 5.4.1] *Let $P : \mathcal{C} \to \{0,1\}$ be any semantic property computable by some analyst $A' \in \mathcal{A}$. Then there is a **black-box** analyst $A \in \mathcal{A}$ such that for every s and every circuit $C \in \mathcal{C}$ of size s on n inputs, $A^C(1^n 0^{s-n}) = P(C)$.*

In prior work, the class of analysts $\mathcal{A}$ was always set to be BPP, and the class of circuits $\mathcal{C}$ was generally set to be unrestricted circuits of fan-in two. In that full form, proving the BBH would also prove $\mathsf{NP} \not\subseteq \mathsf{BPP}$, so that is presently out of reach! (The BBH could also end up being false, of course.) By considering a range of natural possible choices for the weak analysts $\mathcal{A}$ and the circuit sets $\mathcal{C}$, we can try to delineate precisely how weak the analysts from $\mathcal{A}$ need to be, in order for $\mathcal{C}$-circuits to *provably*

behave like black boxes, and to relate the corresponding Black-Box Hypotheses to other core problems within complexity.

5.2 Overview

We demonstrate several interesting relationships between circuit lower bounds and Black-Box Hypotheses in the generalized setting. First, we prove that certain instances of the Black-Box Hypothesis are true, from known circuit lower bounds. In fact we give a generic connection from lower bounds to Black-Box Hypotheses. We also give some converse results, showing that Black-Box Hypotheses imply certain circuit lower bounds. Finally, in some settings, we can show that certain problems are "complete" for a Black-Box Hypothesis, in the sense that proving the Black-Box Hypothesis is *equivalent* to proving a lower bound against the aforementioned problem.

5.2.1 Black-Box Hypotheses For Restricted Analysts, From Lower Bounds

In Section 5.5, we explore situations in which known lower bounds imply Black-Box Hypotheses. We first consider Hypothesis 5.4.1 where the classes of analysts $\mathcal{A}$ are restricted, and the set of potential "boxes" $\mathcal{C}$ consists of *unrestricted circuits*. We show that one can prove a $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$, when the given set of boxes $\mathcal{C}$ is sufficiently powerful and the set of analysts $\mathcal{A}$ is limited. *We find this to be counterintuitive.* It could have been the case that, when the set of boxes $\mathcal{C}$ is powerful, an analyst with access to the code of such a powerful box might be able to learn something interesting about it, and gain more power than if it only had black-box access. However, it turns out that when the boxes are sufficiently powerful, no analyst can learn any *semantic* property.

We find that, under very general conditions, circuit lower bounds against $\mathcal{A}$ (as an algorithmic class) imply the Black-Box Hypothesis for $\mathcal{A}$ (as an analyst class).

Theorem 5.2.1 [Informal Statement, cf. Theorem 5.5.1] *Let $\mathcal{A}$ be a circuit class (of analysts), and let f be a Boolean function computable with (general) circuits of size at most $t(n)$. Suppose $f \notin \mathcal{A}$, and suppose $\mathcal{A}$ is closed under projections from n variables onto $O(t(n) \log t(n))$ variables. Then the (general) Black-Box Hypothesis for $\mathcal{A}$ is true.*

The full formal version of the theorem appears in Section 5.5 as Theorem 5.5.1. Intuitively, we apply a "input-switching" trick which reduces the task of computing f on an input $\mathbf{y}$ to the task of deciding *any* non-trivial semantic property P on a circuit $D_{\mathbf{y}}$.[1] In particular, given an analyst A computing P, we show how to map every Boolean string $\mathbf{y}$ (a potential input for f) into a circuit $D_{\mathbf{y}}$ whose input-output behavior (and in particular, whether $D_{\mathbf{y}}$ satisfies the property P) depends on the value $f(\mathbf{y})$. At a high level, $D_{\mathbf{y}}$ takes an input $\mathbf{x}$, evaluates $f(\mathbf{y})$, and then (depending on $f(\mathbf{y})$) evaluates and outputs either $C_1(\mathbf{x})$ or $C_2(\mathbf{x})$, where C_1 and C_2 are fixed circuits (independent of $\mathbf{y}$), exactly one of which satisfies the property P. In essence, we are "switching" the input $\mathbf{y}$ with a circuit $D_{\mathbf{y}}$ which can evaluate f, and for which we can determine P. Then, we can run A on $D_{\mathbf{y}}$ *without ever evaluating f directly*, and use its answer to determine $f(\mathbf{y})$.

The conditions we impose on $\mathcal{A}$ are quite general, so Theorem 5.5.1 has several direct corollaries. For example, recall from Definition 2.2.15 that AC^0 is the class of unbounded fan-in circuits of constant depth over AND, OR, and NOT.

Corollary 5.2.1 *The BBH for (polynomial-size) AC^0 analysts is true. Moreover, the BBH for $2^{n^{o(1)}}$-size AC^0 is true.*

In particular, Theorem 5.5.1 implies that for every subexponential-size AC^0 circuit family $\{A_n\}$ that is given the code of an arbitrary (general) circuit C as input, if $\{A_n\}$ computes a semantic property (i.e., its output depends only on the function computed

[1] At this level of generality, the idea is similar in spirit to one of the proofs of Rice's Theorem [92] which shows that any non-trivial semantic property of Turing machines is undecidable, by way of a reduction from the Halting Problem. However, Rice's proof techniques do not translate to finite circuits, so we prove Theorem 5.5.1 differently.

by C, not the code of C) then $\{A_n\}$ must compute a trivial property (all-zeroes or all-ones). Similarly:

Recall from Definition 2.2.20 and Notation 2.2.3 that TC^0_2 is the class of unbounded fan-in circuits of depth-two over MAJORITY, AND, OR, and NOT.

Corollary 5.2.2 *The BBH for (polynomial-size)* TC^0_2 *is true. Moreover, the BBH for* $2^{n^{1-\varepsilon}}$*-size* TC^0_2 *analysts is true for every* $\varepsilon > 0$.

5.2.2 A Generalization

Next, we turn to an even more general setting of Black-Box Hypotheses, where both the class $\mathcal{A}$ of "analysts" and the set $\mathcal{C}$ of "boxes" can vary. Here we find that, roughly speaking, if $\mathcal{A}$ and $\mathcal{C}$ jointly satisfy some natural closure properties, and there are functions computable by boxes in $\mathcal{C}$ but not by analysts in $\mathcal{A}$, then the $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ still holds.

Theorem 5.2.2 [Informal Statement, cf. Theorem 5.5.2 and Theorem 5.5.3] *Let* $\mathcal{A}$ *be a circuit (analyst) class, let* $\mathcal{C}$ *be a set of circuits, and let* $f \notin \mathcal{A}$ *be a Boolean function. Suppose there is an analyst in* $\mathcal{A}$ *which, given input* $\mathbf{y}$*, generates a circuit* $D_{\mathbf{y}} \in \mathcal{C}$ *whose input-output behavior depends on the value of* $f(\mathbf{y})$*. Then the* $\mathcal{C}$*-Black-Box Hypothesis for* $\mathcal{A}$ *is true.*

We prove two formal versions of this theorem in Section 5.5.1, as Theorem 5.5.2 and Theorem 5.5.3. These theorems are general enough that $\mathcal{C}$ does not *have* to be a class of circuits *per se*: other non-uniform computational models, such as branching programs or span programs, would also work. The intuition and proof techniques are similar to those used in Theorem 5.5.1, but given the extra conditions on $\mathcal{A}$, we can tailor the input-switching reduction from Theorem 5.5.1 to the set $\mathcal{C}$ in order to produce stronger results. For example:

Recall from Definition 2.2.18 that $\mathsf{AC}^0[p]$ is the class of unbounded fan-in circuits of constant depth over AND, OR, MOD_p, and NOT.

Corollary 5.2.3 *For all primes p, the $\mathsf{AC}^0[p]$-Black-Box Hypothesis for (poly-size) AC^0 holds.*

Theorem 5.5.3 implies that for every AC^0 circuit family $\{A_n\}$ that tries to analyze the code of a given $\mathsf{AC}^0[p]$ circuit C, if $\{A_n\}$ computes a semantic property of C, then that property must be trivial. More generally, we can conclude the following.

Theorem 5.2.3 *For all depths $d \geq 2$ and all distinct primes $p \neq q$, the $\mathsf{AC}^0_d[p]$-Black-Box Hypothesis for $2^{s^{o(1)}}$-size $\mathsf{AC}^0[q]$ analysts is true.*

That is, even if in the above, $\{A_n\}$ can have subexponential size, use $\textsc{Mod}_q$ gates, and fail on input circuits C with depth greater than a fixed constant d, $\{A_n\}$ must *still* compute a trivial property. Similarly:

Theorem 5.2.4 *For all depths $d \geq 2$, the AC^0_d-Black-Box Hypothesis for $2^{s^{o(1)}}$-size AC^0_{d-1} analysts is true.*

5.2.3 Equivalences With Lower Bounds?

So far, our results show how lower bound statements of the form $\mathcal{C} \not\subset \mathcal{A}$ can sometimes be applied to prove the corresponding $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ analysts. A natural next question is, could Black-Box Hypotheses (for various pairs of boxes and analysts) be *equivalent* to proving lower bounds? As a first step, in Section 5.6 we prove conditional lower bounds against some analyst classes $\mathcal{A}$, assuming some $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$.

Theorem 5.2.5 [Informal Statement, cf. Theorem 5.6.1] *Suppose every analyst in $\mathcal{A}$ has subexponential-size circuits, and let $\mathcal{C}$ be a "reasonable" set of circuits (left undefined here). If the $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ is true, then the circuit satisfiability problem for $\mathcal{C}$-circuits is not in $\mathcal{A}$.*

Roughly speaking, we observe that if the $\mathcal{C}$-circuit Evaluation problem ($\mathcal{C}$-Eval) is not in $\mathcal{A}$, then the $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ is **true**, and if the $\mathcal{C}$-circuit

Satisfiability problem ($\mathcal{C}$-SAT) is in $\mathcal{A}$, then the $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ is **false**. However, $\mathcal{C}$-SAT is generally harder than $\mathcal{C}$-EVAL.

To better understand how lower bounds connect to Black-Box Hypotheses, we propose a notion of *BBH-completeness* for computational problems. Very roughly, we want a $\mathcal{C}$-BBH-complete problem Π to have the property that $\Pi \in \mathcal{C}$, and for a general analyst class $\mathcal{A}$, if $\Pi \notin \mathcal{A}$ then the $\mathcal{C}$-BBH for $\mathcal{A}$ is true. We show that for *nondeterministic* circuit classes $\mathcal{C}$, both $\mathcal{C}$-SAT and $\mathcal{C}$-EVAL are $\mathcal{C}$-BBH-complete.

Theorem 5.2.6 [Informal Statement, cf. Theorem 5.6.2] *Suppose every analyst in $\mathcal{A}$ has subexponential-size circuits, and let $\mathcal{C}$ be a nondeterministic circuit class with "natural" closure properties. Then $\mathcal{C}$-EVAL and $\mathcal{C}$-SAT are both $\mathcal{C}$-BBH-complete for $\mathcal{A}$.*

Theorem 5.6.2 shows that lower bounds for the satisfiability problem are equivalent in some sense to proving that nondeterministic circuits behave like black boxes. Impagliazzo, Kabanets, Kolokolova, McKenzie, and Romani [65] considered the question of whether one can show the Black-Box Hypothesis is equivalent to $\mathsf{NP} \not\subset \mathsf{P/poly}$, with some partial results. A consequence of Theorem 5.6.2 is that $\mathsf{NP} \not\subset \mathsf{P/poly}$ *is* equivalent to the Black-Box Hypothesis when polynomial-size circuits are the analysts and *nondeterministic circuits* are the boxes. In this light, it would be very interesting if one could show the Black-Box Hypothesis is actually equivalent to $\mathsf{NP} \not\subset \mathsf{P/poly}$: it would show that two rather different-looking forms of the Black-Box Hypothesis are in fact equivalent.

Finally, we note that the aforementioned work of Impagliazzo *et al.* on BBH [65, 93] yields another kind of equivalence between a different variant of black-box hypothesis and a circuit lower bound.

Theorem 5.2.7 (Follows from [65]) [informal, cf. Theorem 5.6.3] *The following are equivalent:*

1. *The Circuit Satisfiability problem,* CKT-SAT*, is not in* $\mathsf{P/poly}$*.*

2. *Any* symmetric *property P that can be decided in* $\mathsf{P}/poly$ *with white-box access to the input circuit can also be decided in* $\mathsf{P}/poly$ *with black-box access to the input circuit.*

We view this interpretation of their result as further promising evidence towards more general connections between black-box hypotheses and circuit lower bounds.

5.2.4 Organization

Section 5.3 covers significant prior work related to black-box hypotheses. Section 5.4 carefully discusses how to generalize the Black-Box Hypothesis for various sets of "analysts" and sets of "boxes". Section 5.5 proves our main theorems, showing how circuit lower bounds imply Black-Box Hypotheses in a very generic way. Section 5.6 considers how we might prove equivalences between Black-Box Hypotheses and lower bounds. Section 5.7 concludes.

5.3 Background

We study generic versions of the circuit evaluation and satisfiability problems. In the following, let $\mathcal{C}$ be a set of circuits.

Definition 5.3.1 ($\mathcal{C}$-Eval) *In the $\mathcal{C}$-Eval Problem $\mathcal{C}$-*Eval*, we receive a circuit $C \in \mathcal{C}$ and a string x as input, and we must output 1 if and only if C accepts x.*

Definition 5.3.2 ($\mathcal{C}$-Sat) *In the $\mathcal{C}$-Sat Problem $\mathcal{C}$-*Sat*, we receive a circuit $C \in \mathcal{C}$ as input, and we must output 1 if and only if C does not implement the constant zero function.*

Definition 5.3.3 (Ckt-Eval) *The* Circuit Evaluation Problem Ckt-Eval *is $\mathcal{C}$-*Eval *with $\mathcal{C}$ taken to be the set of all Boolean circuits.*

Definition 5.3.4 (Ckt-Eval) *The* Circuit Satisfiability Problem Ckt-Sat *is $\mathcal{C}$-*Sat *with $\mathcal{C}$ taken to be the set of all Boolean circuits.*

Historically, researchers interested in so-called "black-box hypotheses" were looking for what they called a "scaled-down" Rice's Theorem. In the following paragraphs, we provide a brief overview of this research.

5.3.1 Rice's Theorem

We briefly recall the statement and implications of Rice's Theorem. Let $\mathcal{M}$ be the set of Turing Machines.

Definition 5.3.5 (Semanticity) *A property $P : \mathcal{M} \to \{0,1\}$ of Turing Machines is* semantic *if $P(M)$ depends only on the language accepted by M.*

That is, for any TMs M_1 and M_2 that accept the same set of strings, $P(M_1) = P(M_2)$.

Definition 5.3.6 (Non-triviality) *A property P is* non-trivial *if there are $M_1, M_2 \in \mathcal{M}$ such that $P(M_1) \neq P(M_2)$.*

In his 1951 doctoral thesis, Henry Rice proved the following sweeping result:

Theorem 5.3.1 ([92]) *Every non-trivial semantic property of Turing Machines is undecidable.*

Rice's powerful theorem states that any interesting property that we might want to test of a given program is undecidable, assuming the property being tested depends only on the *function computed by the program.* That is, any property that could in principle be tested using only black-box access to the program, is undecidable given a *description* the program. Rice's theorem generalizes (and can be proved from) the undecidability of the TM-SAT problem of determining whether a given Turing Machine accepts any string at all.

5.3.2 The Black Box Hypothesis

In their pioneering obfuscation work, Barak et al. [14] consider the question: *can Rice's Theorem be scaled down in a way that would be useful to complexity theory?*

Specifically, let us assume we are not interested in *all* Turing Machines, but rather in the set of efficient algorithms; for example, those represented by Boolean circuits. One can still define properties that are non-trivial and semantic when restricted to the set of Boolean circuits. In this setting, all such properties P are decidable, because the language of a circuit is simply its 2^n-bit truth table, which can be computed in finite time. However, one might want to know something about the *computational complexity* of such properties. In this setting, the circuit satisfiability problem Ckt-Sat is an analogue of TM-Sat. Although Ckt-Sat is decidable, it is NP-hard, so one might hope to be able to replace undecidability in Rice's Theorem with NP-hardness.

In earlier work, Borchert and Stephan [27] note that using circuits instead of Turing Machines and NP-hardness instead of undecidability is not enough to prove an analogue of Rice's Theorem. For every string x, the property $\{M \in \mathcal{M} : M(x) = 1\}$ is undecidable by Rice's Theorem, but the circuit analogue is decidable in polynomial time: it is simply the circuit evaluation problem! Borchert and Stephan's response to this issue is to look at function properties depending on more complex measures, such as the *number* of Sat assignments of a given circuit (in other words, the property is a symmetric Boolean function in the truth table of the circuit). They show that any non-trivial "counting" property of circuits is UP-hard; the UP-hardness was improved in [62].

Barak *et al.* [14] gave a different response to the above issue. They observe the property $\{C : C(x) = 1\}$ for circuits C is still "trivial" in some sense: it can be efficiently determined given only *black-box* oracle access to the input circuit. This observation led Barak *et al.* to formulate the following conjecture. For two circuits C and C' on n-bit inputs, we write $C \equiv C'$ when C and C' compute the same n-bit function.

Conjecture 5.3.1 (Black Box Hypothesis [14]) *Suppose $L \subseteq \{0,1\}^*$ satisfies the property that for all C and C' such that $C \equiv C'$, we have $C \in L \iff C' \in L$. If $L \in$ BPP, then there is a probabilistic polynomial time algorithm S that decides L*

given only oracle access to C *and* $0^n 1^{|C|-n}$ *as input, i.e.,*

$$C \in L \implies \Pr\left[S^C\left(0^n 1^{|C|-n}\right) = 1\right] > \frac{2}{3}$$

$$C \notin L \implies \Pr\left[S^C\left(0^n 1^{|C|-n}\right) = 1\right] < \frac{1}{3}$$

That is, the BBH claims that every "white box" semantic property of circuits that is decidable in randomized polynomial time can also be decided in randomized polynomial time with "black box" access to the circuit. If the conjecture were true, then a strong form of $\mathsf{P} \neq \mathsf{NP}$ would follow: $\mathsf{P} = \mathsf{NP}$ implies that circuit satisfiability is solvable in polynomial-time when we have "white-box" access to the input circuit, but the SAT problem requires $\Omega(2^n)$ time to solve with only black-box oracle access to the input circuit.

Impagliazzo *et al.* [65] proved interesting results towards understanding BBH. They show a partial converse of the observation from the previous paragraph: if the BBH is false for certain kinds of properties, then the circuit satisfiability problem has sub-exponential size circuits. Since we know that BBH implies $\mathsf{P} \neq \mathsf{NP}$, this suggests that it may be difficult to resolve BBH regardless of its truth or falsity. Romani's master thesis [93] gives an excellent overview of the BBH and this work.

5.3.3 Obfuscation in Cryptography

In recent years, the theory of program obfuscation has exploded into a huge subject area within cryptography, starting with the influential paper of Barak *et al.* [14] which crystallized several key definitions and proved key impossibility results for obfuscation. Two major concepts they proposed are *virtual black-box obfuscation* (VBB for short) and *indistinguishability obfuscation* (iO for short), which we now describe briefly.

A VBB obfuscator $\mathcal{O}$ would take any efficient program/circuit C of size s, and output the code of an "obfuscated" $\mathcal{O}(C)$ such that, for every probabilistic polynomial time (PPT) adversary A, there is another PPT adversary A', such that the probability A outputs 1 on the input $\mathcal{O}(C)$ is very close to the probability that A' outputs 1 on

$(1^n, 1^s)$ when *given C as an oracle*. That is, whatever computation A is doing on the code of $\mathcal{O}(C)$, A' can simulate that knowing only the size of C, its number of inputs, and with input-output access to C. Barak *et al.* showed that there are tasks for which VBB obfuscation is *impossible* assuming one-way functions exist. The notion of iO asks for a weaker guarantee: for all PPT A, and all pairs of size-s circuits C_1, C_2 such that $C_1 \equiv C_2$, the probability A outputs 1 on C_1 is very close to the probability A outputs 1 on C_2. In contrast to VBB, iO is possible under plausible hardness conjectures (e.g., [48, 49, 23]), and it turns out to be very powerful, capable of implementing deniable encryption, public-key encryption from one-way functions, multiparty key exchange, and more (e.g., [94, 26]).

All of the above work on building obfuscation requires hardness assumptions that are unproven (and are typically much stronger than $\mathsf{P} \neq \mathsf{NP}$), and study how we might efficiently transform arbitrary code into obfuscated code, relative to some class of adversarial analysts.

We briefly note the connection between VBB and the BBH. One can think of a VBB obfuscator as an efficient mapping from general circuits to "obfuscated class of circuits", a *restricted subclass* of circuits, such that the BBH holds when the analyzable code $\mathcal{C}$ must come from this restricted subclass. Namely, the VBB property says that, for any efficient analyst that takes circuits from this class as input, there is an efficient black-box analyst that can carry out essentially the same analyses. That is, when VBB is possible, there is a "promise" class of circuits (the image of the obfuscator) for which a black-box hypothesis is true. Accordingly, Barak *et al.* [14] showed that a "promise" version of the BBH is false, assuming one-way functions exist.

5.3.4 Automated Formal Verification

Additionally, settings in which the Black-Box Hypothesis is false are of great interest in automated formal verification. One central question is the following: what properties of a program's input-output behavior can be more efficiently tested by analyzing the program's code, than by treating it as a black box and simply running

it on selected inputs? Many properties of interest depend on the program's behavior on all possible inputs, which may be infeasible (or even impossible) to determine exhaustively. One may instead want to analyze the code of the program in order to determine whether or not it satisfies the given property. This may still be impossible, as many properties of interest are Turing-complete when considered over the space of all possible programs. However, by restricting the class of programs being tested, some such verification problems can become feasible, cf. [87, 88, 6]. In fact, in any setting where the class of programs being analyzed is restricted such that the black box hypothesis is *false*, there *must* exist properties that can be tested by analyzing the program but not by treating it as a black box.

5.4 Generalized Black-Box Hypotheses

We study the Black-Box Hypothesis (Conjecture 5.3.1) in a more general setting. Specifically, instead of considering $L \in \mathsf{BPP}$ and a randomized uniform algorithm S from Conjecture 5.3.1, we study the family of hypotheses that arise when L and S come from various (possibly non-uniform) circuit classes, which may be weaker or stronger than probabilistic poly-time.

Let us first set up some notation.

Notation 5.4.1 (Circuit Description) *For a circuit C, let $\langle C \rangle$ denote the* binary description *of C.*

Note that if C has size s, then $\langle C \rangle$ is a binary string of length $O(s \log s)$, which we call the *description length* of C.

Let $\mathcal{C}$ be a set of circuits.

Definition 5.4.1 (Circuit Property) *A* property *of circuits in $\mathcal{C}$ is a function $P : \mathcal{C} \to \{0, 1\}$.*

Definition 5.4.2 (Semanticity) *A property P of circuits in $\mathcal{C}$ is* semantic *if and only if for any two circuits $C_1, C_2 \in \mathcal{C}$ computing the same function (that is, $\forall \mathbf{x}, C_1(\mathbf{x}) =$*

$C_2(\mathbf{x})$*),* $P(C_1) = P(C_2)$*.*

Recall that a circuit family is an infinite sequence of circuits, one for each possible input length; circuit families compute functions of the form $f : \{0,1\}^\star \to \{0,1\}$ in the natural way.

Definition 5.4.3 *A circuit family* $\{A_s\}$ computes P *if for every circuit* $C \in \mathcal{C}$ *with description length* s*,* $A_s(\langle C \rangle) = P(C)$*.*

For the remainder of this chapter, we will define a circuit class $\mathcal{A}$ to be a set of circuit families (rather than a set of functions); our analyst classes $\mathcal{A}$ will have this form. By convention, an *oracle circuit* C may have oracle gates of arbitrary fan-in, but we will think of C as taking an oracle O with a fixed number of inputs. If C contains oracle gates with a different number of inputs than the given oracle O, then we define such oracle gates to output the constant 0 (regardless of O).

We formulate a generalization of the Black Box Hypothesis, which we call the $\mathcal{C}$-Black Box Hypothesis for $\mathcal{A}$ ($\mathcal{C}$-BBH for $\mathcal{A}$), in the following way.

Hypothesis 5.4.1 (Generalized Black Box Hypothesis: $\mathcal{C}$-BBH for $\mathcal{A}$) *Let* P *be a semantic property of circuits in* $\mathcal{C}$*. Let* $\{A'_s\} \in \mathcal{A}$ *be a circuit family that computes* P*. Then there exists a circuit family* $\{A_s\} \in \mathcal{A}^{\mathcal{C}}$ *such that* $A_s^C(1^n 0^{s-n}) = 1$ *iff* $P(C) = 1$*.*

That is, the $\mathcal{C}$-BBH for $\mathcal{A}$ hypothesizes that every semantic property of $\mathcal{C}$-circuits that can be decided by $\mathcal{A}$-analysts with "white box" access to the $\mathcal{C}$-circuit, can also be decided by $\mathcal{A}$-analysts with only black-box access to the circuit. When $\mathcal{C}$ is the set of all Boolean circuits, we refer to the $\mathcal{C}$-BBH for $\mathcal{A}$ simply as the "BBH for $\mathcal{A}$". Note that if we replace $\mathcal{A}$ in the above with BPP, we recover Conjecture 5.3.1.

5.4.1 Encoding Circuits

Unfortunately, if we allow the class of analysts $\mathcal{A}$ to be an arbitrary circuit class, we can encounter some strange (and counterintuitive) consequences. For instance,

suppose $\mathcal{A}$ is AC^0, the circuit families over AND, OR, and NOT with constant-depth, polynomial size, and unbounded fan-in. We can construct an oracle circuit family $\{A_s\}$ such that $A_s^C(1^n 0^{s-n}) = \textsc{Parity}(n)$, the parity of the number of inputs of C (A_s^C ignores C, and just computes the parity of strings of the form $1^\star 0^\star$). Depending on how the description $\langle C \rangle$ is represented, this behavior may not be computable by *any* white-box AC^0 circuit family $\{A'_s\}$, since PARITY is not in AC^0 [2, 47]! We would like to avoid this sort of behavior, because as in Conjecture 5.3.1, the oracle circuit family A is supposed to capture some notion of *triviality*. In order for the "BBH for $\mathcal{A}$" to be meaningful, it should be that the white-box circuit family A' is at least as powerful as the black-box family A. To this end, we shall require the binary descriptions of circuits to contain all the information given freely to the oracle family. Specifically, we assume that the description of a circuit C with n input wires is prefixed by $1^n 0$, and that the first n wires in $\langle C \rangle$ are the input wires.

5.5 Circuit Lower Bounds Imply Black-Box Hypotheses

What can we prove about the BBH for general pairs of circuit sets and analysts $\mathcal{C}$, $\mathcal{A}$? First, we can show there are interesting pairs for which the $\mathcal{C}$-BBH for $\mathcal{A}$ is true in a strong way: every semantic property is in fact trivial. The following theorem shows that, whenever lower bounds hold against a circuit class $\mathcal{A}$ satisfying some simple conditions, the (general) BBH for $\mathcal{A}$ is true. First, we recall a definition.

Definition 5.5.1 (Projection) *A* projection *from n variables onto m variables is a function $\pi : \{0,1\}^n \to \{0,1\}^m$ such that for every j, there exists i such that the j^{th} coordinate of $\pi(\mathbf{x})$ depends only on the i^{th} coordinate of $\mathbf{x}$.*

Observe that a projection is a kind of very weak reduction which can be computed not only very efficiently but also very *locally*. By requiring closure under such a weak class of reductions, we aim to keep $\mathcal{A}$ as general as possible.

Theorem 5.5.1 *Let $\mathcal{A}$ be a circuit class, $f : \{0,1\}^\star \to \{0,1\}$ be a decision problem, and $s : \mathbb{N} \to \mathbb{N}$ be a monotone function with the properties:*

1. *f is computable by a size-$s(n)$ circuit family, but f is not computable by any family in $\mathcal{A}$.*

2. *Either $\{\text{OR}_n \circ \text{AND}_2\} \subseteq C \in \mathcal{A}$ for some family C, or $\{\text{AND}_n \circ \text{OR}_2\} \subseteq C \in \mathcal{A}$ for some family C. That is, either $\mathcal{A}$ contains a family that either computes the read-once n-clause 2-DNFs on $2n$ variables, or it contains a family that computes the n-clause 2-CNFs on $2n$ variables.*

3. *$\mathcal{A}$ is closed under composition with projections from n variables onto $O(s(n) \log s(n))$ variables.*

Then for every property P over the set of all circuits, if P is semantic and computable in $\mathcal{A}$, then for all n, P restricted to circuits on n-bit inputs is also trivial. In particular, the (general) BBH for $\mathcal{A}$ is true.

Proof. Let $\mathcal{A}$ and f satisfy the above properties, and let $\{F_n\}$ be a size-$s(n)$ circuit family computing f. Let P be a semantic property computable in $\mathcal{A}$.

First, we will prove that P is trivial. The idea is that, if P is not trivial, we can use a circuit family for P to construct a circuit in $\mathcal{A}$ for computing f, a contradiction to the assumed lower bound on f (assumption 1).

Let $k \in \mathbb{N}$. P is semantic, so assume WLOG that for every k-input circuit K_0 computing the constant 0 function, $P(K_0) = 0$. Assume for sake of contradiction that there is a k-input C_k such that $P(C_k) = 1$. Let $n \in \mathbb{N}$ be our desired input length; we want to build a circuit computing f on n-bit inputs. For an n-bit vector $\mathbf{y}$, define the following circuit $D_\mathbf{y}$ with k input wires $\mathbf{x}$, with $\mathbf{y}$ hard-coded as n constant wires:

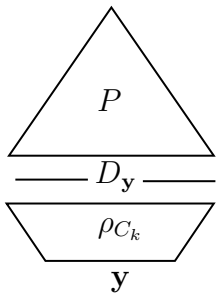
$$D_\mathbf{y}(\mathbf{x}) := C_k(\mathbf{x}) \wedge F_n(\mathbf{y}).$$

The circuit $D_\mathbf{y}$ computes some Boolean function on k input bits. For a fixed C_k, define the function ρ_{C_k} that maps the n-bit input $\mathbf{y}$ to the description $\langle D_\mathbf{y} \rangle$ of $D_\mathbf{y}$

as defined above. Observe that for all $\mathbf{x}$ and $\mathbf{y}$, $D_{\mathbf{y}}(\mathbf{x}) = C_k(\mathbf{x})$ if $f(\mathbf{y}) = 1$, and otherwise $D_{\mathbf{y}}(\mathbf{x}) = 0$. Because P is semantic, $P(D_{\mathbf{y}}) = P(C_k) = 1$ if $f(\mathbf{y}) = 1$, and $P(D_{\mathbf{y}}) = 0$ otherwise. In other words, we have $P(D_{\mathbf{y}}) = f(\mathbf{y})$ for all $\mathbf{y}$.

Note the size of $D_{\mathbf{y}}$ is $t(k) + ||C_k|| + 1$, where $||C_k||$ denotes the size of C_k (which is independent of n), so $D_{\mathbf{y}}$ has description length $O(s(n) \log s(n))$. For a fixed C_k, $\rho_{C_k}(\mathbf{y}) = \langle D_{\mathbf{y}} \rangle$ depends only the n-bit vector $\mathbf{y}$. In particular, within the description $\langle D_{\mathbf{y}} \rangle$, the descriptions $\langle C_k \rangle$ and $\langle F_n \rangle$ are both independent of $\mathbf{y}$, so the only bits in $\langle D_{\mathbf{y}} \rangle$ that vary with $\mathbf{y}$ are those describing the hard-coded constant $\mathbf{y}$ itself. Hence each bit in $\langle D_{\mathbf{y}} \rangle$ depends on at most one bit of $\mathbf{y}$. That is, ρ_{C_k} is a projection from n variables onto $O(s(n) \log s(n))$ variables.

Since $\mathcal{A}$ is closed under such projections (assumption 3), and P is computable in $\mathcal{A}$ by assumption, the circuit



is also computable in $\mathcal{A}$. However, $P(D_{\mathbf{y}}) = f(\mathbf{y})$, which is not computable in $\mathcal{A}$, a contradiction. It follows that for all C_k on k inputs, $P(C_k) = 0$, so P (on circuits containing k inputs) is trivial.

We now turn to proving that there exists an oracle circuit family $\{A_s\}$ in $\mathcal{A}$ such that for any circuit C of size s on n inputs, $A_s^C(1^n 0^{s-n}) = P(C)$. In fact we prove the stronger claim that there exists a circuit family $\{A_s\}$ in $\mathcal{A}$ (with no oracle gates) such that for any circuit C of size s on n inputs, $A_s(1^n 0^{s-n}) = P(C)$. To this end, let $X = \{n \in \mathbb{N} : \exists C \text{ on } n \text{ inputs with } P(C) = 1\}$. First, suppose that $\mathcal{A}$ contains a family that can compute $\{\text{OR}_n \circ \text{AND}_2\}$. For $s \in \mathbb{N}$, let A_s be the circuit of the form

$$\bigvee_{i \in X \cap [s]} (x_i \wedge \neg x_{i+1}) .$$

By assumptions 2 and 3 (closure under projections from n to $2n$ variables), such circuits are in $\mathcal{A}$. If instead $\mathcal{A}$ contains $\{\text{AND}_n \circ \text{OR}_2\}$, we let A_s be the circuit of the form

$$\bigwedge_{i \in [s] \setminus X} (\neg x_i \vee x_{i+1}) .$$

Now $A_s(1^n 0^{s-n}) = 1$ iff $n \in X$ (using no oracle gates). Since P is trivial, for all circuits C on n inputs, $A_s(1^n 0^{s-n}) = 1$ iff $P(C) = 1$, as desired. □

The above proof can be thought of as an "input-switching" trick. We start with the fact that P is non-trivial on some k-bit input circuits. We use the description of a k-input circuit witnessing non-triviality, along with the description of a circuit computing f on n-bit inputs, to construct the description of a larger circuit $D_{\mathbf{y}}$ with n "free variables" $\mathbf{y}$. By feeding n-bit $\mathbf{y}$ into that description, and feeding that description into P, we obtain the description of an $\mathcal{A}$-circuit computing f.

Theorem 5.5.1 has many immediate corollaries. For example:

Reminder of Corollary 5.2.1 *The BBH for (polynomial-size)* AC^0 *is true. Moreover, the BBH for* $2^{n^{o(1)}}$*-size* AC^0 *is true.*

Proof. Take $\mathcal{A}$ to be AC^0 and f to be the PARITY function in Theorem 5.5.1, using the fact that PARITY does not have subexponential-size AC^0 circuits [61]. □

Reminder of Corollary 5.2.2 *The BBH for (polynomial-size)* TC^0_2 *is true. Moreover, the BBH for* $2^{n^{1-\varepsilon}}$*-size* TC^0_2 *is true for every* $\varepsilon > 0$*.*

Proof. Take $\mathcal{A}$ to be TC^0_2 and f to be the INNERPRODUCT function (mod 2) in Theorem 5.5.1, using the fact that INNERPRODUCT requires $2^{\Omega(n)}$-size TC^0_2 circuits [9]. □

5.5.1 Generalization

The proof of Theorem 5.5.1 critically relies on the fact that the circuit $D_{\mathbf{y}}$ can be arbitrarily large and complex in comparison to its input. If we restrict $\mathcal{C}$ to contain

only "simple" circuits and allow A'_s to behave arbitrarily on circuits not in $\mathcal{C}$, then we would need to be more careful to ensure that $D_{\mathbf{y}}$ is still in $\mathcal{C}$. By extending the input-switching trick from Theorem 5.5.1, we can restrict the circuit set $\mathcal{C}$ in some interesting ways and still prove the corresponding Black-Box Hypotheses.

Definition 5.5.2 (Input-Switching Function) *Let $\mathcal{C}$ be a set of circuits, and let f and g be Boolean functions. We say that a function $I : \mathcal{C} \times \{0,1\}^* \to \{0,1\}^*$ is an* input-switching function for $\mathcal{C}$ and f *iff for some bit b, for every circuit $C \in \mathcal{C}$ and every Boolean string $\mathbf{y}$, $I(C, \mathbf{y})$ is the description $\langle D_{\mathbf{y}} \rangle$ of a circuit $D_{\mathbf{y}}$ with the same number of inputs as C such that $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ when $f(\mathbf{y}) = b$ and $D_{\mathbf{y}}(\mathbf{x}) = g(\mathbf{x})$ otherwise.*

Theorem 5.5.2 *Let $\mathcal{A}$ be a circuit class, $f : \{0,1\}^* \to \{0,1\}$ be a decision problem, and $\mathcal{C}$ be a set of circuits with the properties:*

1. *$\mathcal{A}$ computes neither f nor $\neg f$.*
2. *$\mathcal{A}$ is closed under composition with an input-switching function I for $\mathcal{C}$ and f, in the sense that for every function g computable by a circuit family in $\mathcal{A}$ and for every $C \in \mathcal{C}$, the function $\mathbf{y} \mapsto g(I(C, \mathbf{y}))$ is also computable by a circuit family in $\mathcal{A}$.*

Then for every property P over $\mathcal{C}$, if P is semantic and computable in $\mathcal{A}$, then for all input lengths n, P restricted to circuits on n-bit inputs is also trivial. Furthermore, if $\mathcal{A}$ also contains $\{\mathrm{OR}_n \circ \mathrm{AND}_2\}$ (or $\{\mathrm{AND}_n \circ \mathrm{OR}_2\}$), then the $\mathcal{C}$-BBH for $\mathcal{A}$ is true.

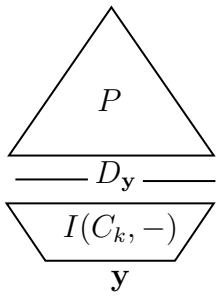
Proof. Let $\mathcal{A}$ and f satisfy the above properties, and let I be the input-switching function for $\mathcal{C}$ and f. Let P be a semantic property computable in $\mathcal{A}$.

First, we will prove that P is trivial. The idea is that, if P is not trivial, we can use a circuit family for P to construct a circuit in $\mathcal{A}$ for computing f, a contradiction to the assumed lower bound on f.

Let $k \in \mathbb{N}$. Assume WLOG that for every circuit G computing g on k inputs, $P(G) = 0$. Assume for sake of contradiction that there is a k-input C_k such that

$P(C_k) = 1$. For an n-bit input $\mathbf{y}$, consider the circuit $D_\mathbf{y} = I(C_k, y)$. Note that $D_\mathbf{y}$ computes some Boolean function on k input bits. From the definition of I, $D_\mathbf{y}(\mathbf{x}) = C_k(\mathbf{x})$ if $f(\mathbf{y}) = b$, and otherwise $D_\mathbf{y}(\mathbf{x}) = G(\mathbf{x})$. Because P is semantic, $P(D_\mathbf{y}) = P(C_k) = 1$ if $f(\mathbf{y}) = b$, and $P(D_\mathbf{y}) = P(G) = 0$ otherwise. In other words, we have $P(D_\mathbf{y}) = b \otimes f(\mathbf{y})$ for all $\mathbf{y}$.

Since $\mathcal{A}$ is closed under composition with $I(C_K, -)$, and P is computable in $\mathcal{A}$ by assumption, the circuit



is also computable in $\mathcal{A}$. However, $P(D_\mathbf{y}) = f(\mathbf{y})$ or $\neg f(\mathbf{y})$, which are not computable in $\mathcal{A}$, a contradiction. It follows that for all C_k on k inputs, $P(C_k) = 0$, so P (on circuits containing k inputs) is trivial.

As in Theorem 5.5.1, there exists a circuit family $\{A_s\}$ in $\mathcal{A}$ (with no oracle gates) such that for any circuit C of size s on n inputs, $A_s(1^n 0^{s-n}) = P(C)$. □

The preconditions for Theorem 5.5.2 are somewhat too restrictive to be applied easily in many cases, so we strengthen it further. To this end, we first define a relation $\sim_n$ on sets of circuits.

Definition 5.5.3 *For sets $\mathcal{C}_1$ and $\mathcal{C}_2$ of circuits, say that $\mathcal{C}_1 \sim_n \mathcal{C}_2$ iff there exist n-input circuits $C_1 \in \mathcal{C}_1$ and $C_2 \in \mathcal{C}_2$ such that $C_1 \equiv C_2$ (that is, C_1 and C_2 compute precisely the same Boolean function).*

The relation $\sim_n$ enables us to more easily reason about semantic properties across several sets of differently structured circuits.

Theorem 5.5.3 *Let $\mathcal{A}$ be a circuit class, $f : \{0,1\}^* \to \{0,1\}$ be a decision problem, $\mathcal{C} = \bigcup_{i \in \mathbb{N}} \mathcal{C}_i$ be a set of circuits, and $I : \mathcal{C} \times \{0,1\}^* \to \{0,1\}^*$ a function with the properties:*

1. *$\mathcal{A}$ computes neither f nor $\neg f$.*

2. *$\mathcal{A}$ is closed under composition with I.*

3. *For all i, the restriction of I to $\mathcal{C}_i \times \{0,1\}^*$ is an input-switching function for $\mathcal{C}_i$ and f.*

4. *For every input size $n \in \mathbb{N}$, the transitive closure of $\sim_n$ on $\{\mathcal{C}_i\}$ is the universal relation on $\{\mathcal{C}_i\}$.*

Then for every property P over $\mathcal{C}$, if P is semantic and computable in $\mathcal{A}$, then for all input lengths n, P restricted to circuits on n-bit inputs is also trivial. Furthermore, if $\mathcal{A}$ also contains $\{\text{OR}_n \circ \text{AND}_2\}$ (or $\{\text{AND}_n \circ \text{OR}_2\}$), then the $\mathcal{C}$-BBH for $\mathcal{A}$ is true.

Proof. Let P be a property over $\mathcal{C}$. Applying Theorem 5.5.2 to $\mathcal{A}$, f, and to each $\mathcal{C}_i$, for all n and all i, the restrictions of P to circuits in each $\mathcal{C}_i$ with n-bit inputs is trivial. Since P is semantic, if $i \sim_n j$, then the restriction of P to circuits in $\mathcal{C}_i \cup \mathcal{C}_j$ with n-bit inputs is also trivial. Finally since the transitive closure of $\sim_n$ is universal, by induction we have that for every n, the restriction of P to circuits in $\mathcal{C}$ with n-bit inputs is trivial. □

5.5.2 Examples

We now define some input-switching functions. First, let f be any function computable by a circuit family $\{F_n\}$, and let $D_{C,\mathbf{y}}$ be the circuit defined as follows, where $\mathbf{x}$ are the input wires and $\mathbf{y}$ are hard-coded as n constant wires:

$$D_{C,\mathbf{y}}(\mathbf{x}) := C(\mathbf{x}) \wedge F_n(\mathbf{y}).$$

If F_n has size $s(n)$, then the map $\mathbf{y} \mapsto \langle D_{C,\mathbf{y}} \rangle$ (where $\langle D_{C,\mathbf{y}} \rangle$ is the description of $D_{C,\mathbf{y}}$) is both an input-switching function and a projection from n variables onto the $O(s(n) \log s(n))$ variables describing $D_{C,\mathbf{y}}$, so we recover Theorem 5.5.1.

Recall that $\mathsf{AC}^0_d[p]$ denotes circuit families of depth d with unbounded fan-in AND, OR, and MOD_p gates.

Reminder of Corollary 5.2.3 *For all primes p, the $\mathsf{AC}^0[p]$-Black-Box Hypothesis for (polynomial-size) AC^0 is true. Moreover, the $\mathsf{AC}^0[p]$-Black-Box Hypothesis for $2^{s^{o(1)}}$-size AC^0 is true.*

Proof. Follows from Theorem 5.5.2. We make use of the fact that the $\textsc{Mod}_p$ function is computable in linear size $\mathsf{AC}^0[p]$ but requires exponential size in AC^0, and that in AC^0 we can mask a given $\mathsf{AC}^0[p]$ circuit with a given $\textsc{Mod}_p$ function.

Let $\mathcal{A}$ be $2^{s^{o(1)}}$-size AC^0, $f = \textsc{Mod}_p$, and $\mathcal{C} = \mathsf{AC}^0[p]$. We now define a circuit $D_{C,\mathbf{y}}$ with the same number of inputs as C as

$$D_{C,\mathbf{y}}(\mathbf{x}) := C(\mathbf{x}) \wedge \textsc{Mod}_p(\mathbf{y}).$$

Then for all $\mathbf{x}$ and $\mathbf{y}$, $D_{C,\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\textsc{Mod}_p(\mathbf{y}) = 1$, and $D_{C,\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Now the map $(C, \mathbf{y}) \mapsto \langle D_{C,\mathbf{y}} \rangle$ is an input-switching function for $\mathcal{C}$ and $\textsc{Mod}_p$. Furthermore, we can think of the map $\mathbf{y} \mapsto \langle D_{C,\mathbf{y}} \rangle$ as a projection from n variables $\mathbf{y}$ onto $\Theta(n \log n)$ variables describing $D_{C,\mathbf{y}}$, so $\mathcal{A}$ is closed under composition with I. Now from Theorem 5.5.2, every semantic property P over $\mathcal{C}$ computable in $\mathcal{A}$ is trivial, so the $\mathcal{C}$-BBH for $\mathcal{A}$ is true. □

If we invoke Theorem 5.5.3 instead of Theorem 5.5.2, we can get an even stronger result.

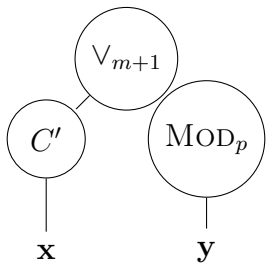
Theorem 5.5.4 *For all depths $d \geq 2$ and distinct primes $p \neq q$, the $\mathsf{AC}^0_d[p]$-BBH for $2^{s^{o(1)}}$-size $\mathsf{AC}^0[q]$ is true.*

Proof. Follows from Theorem 5.5.3. We make use of the fact that the $\textsc{Mod}_p$ function is computable in linear size $\mathsf{AC}^0_d[p]$ but requires exponential size in $\mathsf{AC}^0[q]$ [90, 98], and that in AC^0 we can mask a given $\mathsf{AC}^0_d[p]$ circuit with a given $\textsc{Mod}_p$ function, without increasing its depth.

Let $d \geq 2$, $\mathcal{A}$ be $2^{s^{o(1)}}$-size $\mathsf{AC}^0[q]$, $f = \textsc{Mod}_p$, $\mathcal{C} = \mathsf{AC}^0_d[p]$, $\mathcal{C}_1 = \textsc{Or} \circ \mathsf{AC}^0_{d-1}[p]$, $\mathcal{C}_2 = \textsc{And} \circ \mathsf{AC}^0_{d-1}[p]$, and $\mathcal{C}_3 = \textsc{Mod}_p \circ \mathsf{AC}^0_{d-1}[p]$. (Note that $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3$.) We now define a function $I : \mathcal{C} \times \{0,1\}^* \to \{0,1\}^*$, so that $I(C, \mathbf{y}) = \langle D_{\mathbf{y}} \rangle$, where $D_{\mathbf{y}}$

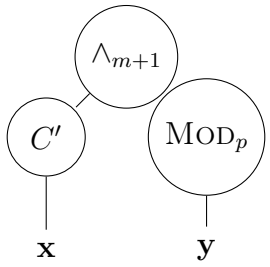
has the same number of inputs as C. We condition on whether the input circuit C comes from $\mathcal{C}_1$, $\mathcal{C}_2$, or $\mathcal{C}_3$.

(Case 1) If $C \in \mathcal{C}_1$, then it has the form $\vee_m \circ C'$, where C' is a depth-$(d-1)$ circuit with n inputs and m outputs, and $\vee_m$ is an OR of fan-in m. For a k-bit vector $\mathbf{y}$, we construct $D_{\mathbf{y}}$ as follows:



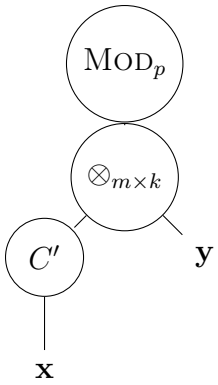
Then for all $\mathbf{x}$ and $\mathbf{y}$, $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\text{MOD}_p(\mathbf{y}) = 0$, and $D_{\mathbf{y}}(\mathbf{x}) = 1$ otherwise. Hence the restriction of I to $\mathcal{C}_1 \times \{0,1\}^*$ is an input-switching function for $\mathcal{C}_1$ and f.

(Case 2) If $C \in \mathcal{C}_2$, we construct $D_{\mathbf{y}}$ similarly to case (1). Assuming $C = \wedge_m \circ C'$ for the same sort of C', we can construct $D_{\mathbf{y}}$ as follows:



Then for all $\mathbf{x}$ and $\mathbf{y}$, $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\text{MOD}_p(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Hence the restriction of I to $\mathcal{C}_2 \times \{0,1\}^*$ is an input-switching function for $\mathcal{C}_2$ and f.

(Case 3) If $C \in \mathcal{C}_3$, then C is a MOD_p gate of fan-in m, composed with some depth-$(d-1)$ circuit C' having n inputs and m outputs. We define a $\otimes_{m \times k}$ gate to take $m+k$ inputs $x_1, \ldots, x_m, y_1, \ldots, y_k$, and output $x_i \cdot y_j$ for all i, j, and define a circuit $D'_{\mathbf{y}}(\mathbf{x})$ as follows:



Note that for C of depth d, $D'_{\mathbf{y}}$ has depth $d+1$. However, when treating $\mathbf{y}$ as a constant, each $C'(\mathbf{x})_i \wedge y_j$ simplifies to a single wire (either $C'(\mathbf{x})_i$ if $y_j = 1$, or the constant 0 if $y_j = 0$). Performing these simplifications and removing the layer of AND gates, we get a circuit $D_{\mathbf{y}}$ of depth d. (Note that each bit in $\langle D_{\mathbf{y}} \rangle$ still only depends on at most one bit of $\mathbf{y}$.) Now for all $\mathbf{x}$ and $\mathbf{y}$, $D_{\mathbf{y}}(\mathbf{x}) = \text{MOD}_p\left(C'(\mathbf{x}) \otimes \mathbf{y}\right) = \text{MOD}_p\left(C'(\mathbf{x})\right) \times \text{MOD}_p(\mathbf{y}) = C(\mathbf{x}) \times \text{MOD}_p(\mathbf{y})$. That is, $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $\text{MOD}_p(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Hence the restriction of I to $\mathcal{C}_3 \times \{0,1\}^*$ is an input-switching function for $\mathcal{C}_3$ and f.

Next, we observe that in every case, each bit in $\langle D_{\mathbf{y}} \rangle$ depends on only one bit of $\mathbf{y}$, so $\mathcal{A}$ is closed under composition with I (a projection). Finally, there are circuits C_1, C_2, C_3, which have an OR, AND, and MOD_p output gate (respectively), yet $C_1 \equiv C_2 \equiv C_3$ (e.g. they can ignore their input and ouput the constant 0). Hence $\sim_k$ as defined in Theorem 5.5.3 is the universal relation. Now from Theorem 5.5.3, every semantic property P over $\mathcal{C}$ computable in $\mathcal{A}$ is trivial, so the $\mathcal{C}$-BBH for $\mathcal{A}$ is true. □

The proof of Theorem 5.5.4 relies on the fact that small $\mathsf{AC}^0[q]$ circuits cannot evaluate some function that can be evaluated with small $\mathsf{AC}^0[p]$ circuits (namely a single MOD_p gate). We can prove a similar result using the depth-d Sipser function, which is easy for AC^0 circuits of depth d but hard for depth $d-1$ [97, 61].

Definition 5.5.4 (Sipser Function) *The* Sipser function $f^{d,n} : \{0,1\}^{\sqrt{\frac{n}{\log n}}} \times \{0,1\}^{n^{d-2}} \times \{0,1\}^{\sqrt{\frac{1}{2}dn\log n}} \to \{0,1\}$ *is defined as follows:*

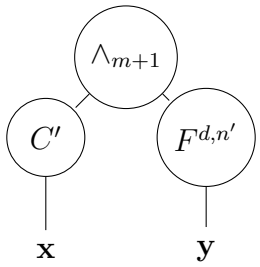
If d is odd, then $f^{d,n}(\mathbf{x}) = \bigwedge_{i_1=1}^{\sqrt{\frac{n}{\log n}}} \bigvee_{i_2=1}^{n} \bigwedge_{i_3=1}^{n} \cdots \bigwedge_{i_d=1}^{\sqrt{\frac{1}{2}dn\log n}} x_{i_1,\ldots,i_d}.$

If d is even, then $f^{d,n}(\mathbf{x}) = \bigwedge_{i_1=1}^{\sqrt{\frac{n}{\log n}}} \bigvee_{i_2=1}^{n} \bigwedge_{i_3=1}^{n} \cdots \bigvee_{i_d=1}^{\sqrt{\frac{1}{2}dn\log n}} x_{i_1,\ldots,i_d}.$

Theorem 5.5.5 *For all depths $d \geq 2$, the AC^0_d-BBH for $2^{s^{o(1)}}$-size AC^0_{d-1} is true.*

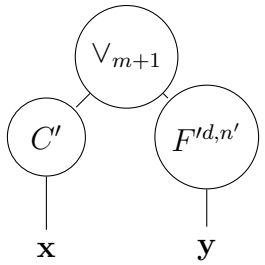
Proof. We proceed as in Theorem 5.5.4. Let $d \geq 2$, f be the depth-d Sipser function, $\mathcal{A}$ be $2^{s^{o(1)}}$-size AC^0_{d-1}, $\mathcal{C} = \mathsf{AC}^0_d$, $\mathcal{C}_1 = \textsc{And} \circ \mathsf{AC}^0_{d-1}$, and $\mathcal{C}_2 = \textsc{Or} \circ \mathsf{AC}^0_{d-1}$. (Note that $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$.) We now define a function $I : \mathcal{C} \times \{0,1\}^* \to \{0,1\}^*$, so that $I(C, \mathbf{y}) = \langle D_{\mathbf{y}} \rangle$, where $D_{\mathbf{y}}$ has the same number of inputs as C. We condition on whether the input circuit C comes from $\mathcal{C}_1$ or $\mathcal{C}_2$.

(Case 1) If $C \in \mathcal{C}_1$, then it has the form $\wedge_m \circ C'$, where C' is a depth-$(d-1)$ circuit with n inputs and m outputs, and $\wedge_m$ is an And of fan-in m. Let $k \in \mathbb{N}$, and take $n' = (2k/d)^{1/(d-1)}$, so that $f^{d,n'}$ has k inputs. For a k-bit vector $\mathbf{y}$, we construct $D'_{\mathbf{y}}$ as follows:



Here, $F^{d,n'}$ denotes the obvious depth-d circuit computing the Sipser function $f^{d,n'}$. Now by collapsing the output And gate of $F^{d,n'}$ into the $\wedge_{m+1}$, we obtain a depth-d circuit $D_{\mathbf{y}}$ on n inputs such that $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $f^{d,n'}(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 0$ otherwise. Hence the restriction of I to $\mathcal{C}_1 \times \{0,1\}^*$ is an input-switching function for $\mathcal{C}_1$ and f.

(Case 2) If $C \in \mathcal{C}_2$, then it has the form $\vee_m \circ C'$. In this case, we construct the circuit $D'_{\mathbf{y}}$ as follows:



Here $F'^{d,n'}$ denotes the circuit obtained by replacing all AND gates in $F^{d,n'}$ with OR gates and vice-versa, and negating all of the input wires. By collapsing the output OR gate of $F'^{d,n'}$ into the $\vee_{m+1}$, we obtain a depth-d circuit $D_{\mathbf{y}}$ on n inputs such that $D_{\mathbf{y}}(\mathbf{x}) = C(\mathbf{x})$ if $f^{d,n'}(\mathbf{y}) = 1$, and $D_{\mathbf{y}}(\mathbf{x}) = 1$ otherwise. Hence the restriction of I to $\mathcal{C}_2 \times \{0,1\}^*$ is an input-switching function for $\mathcal{C}_2$ and f.

As before, we observe that each bit in $\langle D_{\mathbf{y}} \rangle$ depends on at most one bit of $\mathbf{y}$, and that there are circuits C_1 and C_2 which have an AND and OR output gate (respectively) and compute the constant 0 function. Applying Theorem 5.5.3, every semantic property P over $\mathcal{C}$ computable in $\mathcal{A}$ is trivial, so the $\mathcal{C}$-BBH for $\mathcal{A}$ is true. □

5.6 Lower Bounds from Black-Box Hypotheses

In Section 5.5, we showed that many circuit lower bounds of the form $\mathcal{C}' \not\subseteq \mathcal{A}$ can be used to prove a corresponding $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ (for a set of boxes $\mathcal{C}$ that suitably captures the complexity class $\mathcal{C}'$). Now we consider the converse question: can Black-Box Hypotheses also be used to prove circuit lower bounds? For certain sets $\mathcal{C}$ of boxes and classes $\mathcal{A}$ of analysts, it turns out that the $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ does in fact imply lower bounds against $\mathcal{A}$.

For a function $s : \mathbb{N} \to \mathbb{N}$, let CIRCUIT$(s(n))$ denote the set of (general) Boolean circuits on n inputs of size at most $s(n)$, for every n. (Note this is different from SIZE$(s(n))$, which is the class of *languages* computed by circuit families of size at most $s(n)$.) As a starting point, the following simple proposition was essentially noted by Barak *et al.* [14].

Proposition 5.6.1 *If* $\mathsf{NP} \subset \mathsf{P}/poly$*, then for every polynomial* p*, the* CIRCUIT$(p(n))$*-BBH for* $\mathsf{P}/poly$ *is false.*

Proof. Take P to be the CKT-SAT property (that is, $P(C) = 0$ iff the circuit C encodes the all-zeroes function). By assumption, $P \in \mathsf{P}/\text{poly}$, but even with randomness, $\Omega(2^n)$ oracle queries are needed to determine whether a size-$p(n)$ circuit on n inputs is the all-zeroes function. For every polynomial q, the polynomial $q \circ p$ is $o(2^n)$, so there is no size-$q(s)$ circuit family making $\Omega(2^n)$ oracle queries on size-$p(n)$ circuits. □

In fact, Proposition 5.6.1 can be strengthened by replacing CKT-SAT with the property $P(C) = 1$ iff C has a satisfying assignment that sets the first k inputs to 0 (for some appropriately large k).

Proposition 5.6.2 *If* $\mathsf{NP} \subset \text{SIZE}(2^{n^{o(1)}})$*, then for every polynomial* p*, the* CIRCUIT$(p(n))$*-BBH for* $\mathsf{P}/poly$ *is false.*

Propositions 5.6.1 and 5.6.2 are arguably not particularly useful, since very few researchers believe the hypotheses of these propositions. However, they still do illustrate an interesting observation, and we may be able to generalize them in a useful manner. Recall that $\mathcal{C}$-SAT is the satisfiability problem for circuits from the set $\mathcal{C}$. One might hope to prove the following generalization of Proposition 5.6.1, for every circuit set $\mathcal{C}$ and every analyst class $\mathcal{A}$:

Hypothesis 5.6.1 (The Satisfiability Black-Box Hypothesis) *If* $\mathcal{C}$-SAT $\in \mathcal{A}$*, then the* $\mathcal{C}$*-BBH for* $\mathcal{A}$ *is false.*

In this fully generic form, there are some simple counterexamples to Hypothesis 5.6.1. For instance, if $\mathcal{A}$ contains *all* Boolean functions, then (for every set $\mathcal{C}$) $\mathcal{C}$-SAT $\in \mathcal{A}$. However, the $\mathcal{C}$-BBH for $\mathcal{A}$ is *true*, because $\mathcal{A}$ can decide *any* semantic property with only black-box access to the circuit being analyzed. Hence we require additional restrictions on $\mathcal{C}$ and $\mathcal{A}$ to make the hypothesis interesting. In particular, we would like $\mathcal{A}$ to contain only functions of subexponential circuit complexity, and

for a sufficiently simple function f, we would like $\mathcal{C}$ circuits to be able to compute f efficiently.

Recall that a Boolean function $f : \{0,1\}^* \to \{0,1\}$ is a *point function* if there is an $\mathbf{a} \in \{0,1\}^*$ such that for all $\mathbf{x}$, $f(\mathbf{x}) = 1 \iff \mathbf{x} = \mathbf{a}$. The following notion of "reasonability" for circuit sets will be useful in multiple contexts.

Definition 5.6.1 (Reasonability) *A set $\mathcal{C}$ of circuits is* reasonable *if there is a polynomial p such that for all point functions f, there is a circuit family $\{C_n\} \subset \mathcal{C}$ of size at most $p(n)$ computing f.*

We can show that if $\mathcal{C}$ is reasonable and $\mathcal{A}$ has subexponential-size circuits, then Hypothesis 5.6.1 is true. The following can be viewed as a kind of converse of Theorem 5.5.1.

Theorem 5.6.1 *If $\mathcal{C}$ is reasonable, $\mathcal{A} \subseteq \text{SIZE}(2^{n^{o(1)}})$, and $\mathcal{C}$-SAT $\in \mathcal{A}$, then the $\mathcal{C}$-BBH for $\mathcal{A}$ is false.*

Proof. Assume $\mathcal{C}$ is reasonable, $\mathcal{A}$ has subexponential-size circuits, and $\mathcal{C}$-SAT $\in \mathcal{A}$. As in Proposition 5.6.1, we take P to be the satisfiability property. By assumption, $P \in \mathcal{A}$. Even with randomness, $\Omega(2^n)$ oracle queries are required to determine whether a circuit of size $p(n)$ on n inputs computes the constant 0 function. However, an $\mathcal{A}$ circuit can make at most $2^{n^{o(1)}}$ queries to its oracle when given an input of size $p(n)$. Per the reasonableness of $\mathcal{C}$, there are both satisfiable and unsatisfiable $\mathcal{C}$-circuits of size $p(n)$, so $\mathcal{A}$, with only black-box access to a $\mathcal{C}$-circuit, cannot compute P. □

The preconditions for Theorem 5.6.1 are very general; most complexity classes of interest only deal with functions of subexponential complexity and can compute point functions efficiently. However, this weak condition is sufficient to remove the simple counterexamples.

5.6.1 A Notion of BBH-Completeness

For very general circuit sets $\mathcal{C}$ and classes $\mathcal{A}$ of analysts, we have shown (roughly) in Section 5.5 that

$$\mathcal{C} \not\subset \mathcal{A} \implies \mathcal{C}\text{-BBH for } \mathcal{A},$$

and in the previous paragraphs that for "reasonable" $\mathcal{A}$ and $\mathcal{C}$,

$$\mathcal{C}\text{-BBH for } \mathcal{A} \implies \mathcal{C}\text{-SAT} \notin \mathcal{A}.$$

For many pairs of classes $\mathcal{C}$ and $\mathcal{A}$, we have

$$\mathcal{C}\text{-EVAL} \notin \mathcal{A} \iff \mathcal{C} \not\subset \mathcal{A}.$$

So the results of Section 5.5 imply, at least for many natural pairs $\mathcal{C}, \mathcal{A}$, that $\mathcal{C}$-EVAL lower bounds imply BBHs. However, $\mathcal{C}$-SAT is generally a harder problem than $\mathcal{C}$-EVAL, so there remains a gap between the lower bounds that provably imply a Black-Box Hypothesis, and those lower bounds provably implied by a Black-Box Hypothesis.

A natural question is then, which of these implications can be strengthened? **Is there a single problem on $\mathcal{C}$ circuits, such that proving a lower bound for it is equivalent to proving a $\mathcal{C}$-Black-Box Hypothesis?** In particular, is proving either $\mathcal{C}$-EVAL $\notin \mathcal{A}$ or $\mathcal{C}$-SAT $\notin \mathcal{A}$ *equivalent* to proving the $\mathcal{C}$-BBH for $\mathcal{A}$? Similar to other completeness notions in complexity theory, we propose a concept of BBH-completeness to study equivalences between circuit lower bounds and Black-Box Hypotheses.[2]

Definition 5.6.2 (BBH-completeness) *Let $\mathcal{C}$ be a set of circuits and $\mathcal{A}$ a complexity class. A Boolean function f is* complete *for the $\mathcal{C}$-BBH for $\mathcal{A}$ (or $\mathcal{C}$-BBH-complete*

[2]It must be said that the authors are not entirely comfortable with the following definition of BBH-completeness. Ideally, the following would be a consequence of f being BBH-complete, and the actual definition would involve a notion of reducibility. However, in order to give a completeness concept that fits all possible classes $\mathcal{A}$ and $\mathcal{C}$ at a high level of generality, it does not seem possible to use reductions: a sound reducibility notion would inevitably have to depend on $\mathcal{A}$ (in particular, its allowed "sizes" and its closure properties) directly.

for $\mathcal{A}$) iff

$$\mathcal{C}\text{-BBH for } \mathcal{A} \iff f \notin \mathcal{A}.$$

When $\mathcal{A}$ is either implicitly understood or general, we say that f is $\mathcal{C}$-BBH-complete.

Are there natural pairs $(\mathcal{C}, \mathcal{A})$ for which either $\mathcal{C}$-EVAL or $\mathcal{C}$-SAT is $\mathcal{C}$-BBH-complete for $\mathcal{A}$?

5.6.2 Nondeterministic Boxes

For the case of sets $\mathcal{C}$ of nondeterministic circuits, the answer is **yes**. To state our theorem, we require one new concept. Recall that a nondeterministic circuit C has a sequence of "normal" inputs $\mathbf{x}$ as well as a sequence of "auxiliary" nondeterministic inputs $\mathbf{y}$, and we say that C accepts $\mathbf{x}$ if there is a setting of $\mathbf{y}$ such that $C(\mathbf{x}, \mathbf{y}) = 1$.

Definition 5.6.3 (Nondeterminization) *For a given circuit C, a* nondeterminization of C *is a circuit C' in which normal inputs to C have been converted into auxiliary nondeterministic inputs. A set $\mathcal{C}$ of circuits is* closed under nondeterminization *if $C \in \mathcal{C}$ implies that every nondeterminization of C is also in $\mathcal{C}$.*

Theorem 5.6.2 *Let $\mathcal{C}$ be a reasonable set of circuits closed under nondeterminization. Assume $\mathcal{A}$ has circuits of size $2^{n^{o(1)}}$ and that $\mathcal{A}$ is closed under composition with an input-switching function for $\mathcal{C}$ and $\mathcal{C}$-EVAL. Then $\mathcal{C}$-EVAL and $\mathcal{C}$-SAT are $\mathcal{C}$-BBH-complete for $\mathcal{A}$.*

Proof. We wish to prove that the following are equivalent:

1. $\mathcal{C}$-BBH for $\mathcal{A}$
2. $\mathcal{C}$-SAT $\notin \mathcal{A}$
3. $\mathcal{C}$-EVAL $\notin \mathcal{A}$

(1) $\implies$ (2) Follows from Theorem 5.6.1.

(2) $\Longrightarrow$ (3) We reduce $\mathcal{C}$-SAT to $\mathcal{C}$-EVAL by observing that changing the inputs of a nondeterministic circuit into auxiliary nondeterministic inputs preserves satisfiability. Hence, given a nondeterministic circuit C, we can convert *all* of its input bits into additional nondeterministic auxiliary inputs to obtain a circuit C', and then determine whether C' is still satisfiable. However, C' has no remaining free inputs, so determining satisfiability of C' is simply the problem of *evaluating* C' (with no inputs).

(3) $\Longrightarrow$ (1) Follows from Theorem 5.5.2. □

Interpreting Impagliazzo *et al.* as an Equivalence. Recently, Impagliazzo *et al.* [65] proved that if the BBH is false for certain kinds of function properties, then the circuit satisfiability problem has sub-exponential size circuits. In particular, they show that CKT-SAT has $2^{n^{o(1)}}$-size circuits if a property P is highly sensitive on a function f that has sub-exponential size circuits.

Impagliazzo *et al.* indicate that in some sense CKT-SAT is BBH-complete, at least for large analyst classes $\mathcal{A}$. Specifically, if we consider only *symmetric* semantic properties, i.e., properties that depend only on the number of ones in the truth table of the input circuit, we can define the following conjecture:

Hypothesis 5.6.2 (Symmetric-BBH) *Let P be a semantic and* symmetric *property of circuits. Let $\{A'_s\}$ be a polynomial size circuit family. Assume that for every circuit C of size s on n inputs, $A'_s(C) = 1$ iff $P(C) = 1$. Then there exists a polynomial size oracle circuit family $\{A_s\}$ such that $A_s^C(1^n 0^{s-n}) = 1$ iff $P(C) = 1$.*

Now [65] implies:

Theorem 5.6.3 (Follows from [65]) *The following are equivalent:*

1. CKT-SAT *is not in P/poly.*

2. *The Symmetric-BBH is true.*

Proof. The forward direction is Corollary 4.3 in [65]. For the converse direction, observe that CKT-SAT is a symmetric property that requires exponentially many black-box oracle queries (and in particular, cannot be solved in P/poly with only black-box access to the input circuit). Hence if the Symmetric-BBH is true, then CKT-SAT also cannot be solved in P/poly with white-box access to the input circuit, i.e., CKT-SAT $\notin \mathsf{P}$/poly. □

5.7 Conclusion

In this chapter, we introduced *generalized* Black-Box Hypotheses, which parameterize both the type of "box" being analyzed, and the type of "analyst" examining such boxes. We showed that generalized Black-Box Hypotheses can follow generically from circuit lower bounds, and we showed how lower bounds for the circuit satisfiability problem are essentially equivalent to Black-Box Hypotheses where the "boxes" correspond to nondeterministic circuits. We conclude with some additional interesting directions to consider.

5.7.1 What Other Lower Bounds Are Implied by Black-Box Hypotheses?

In Section 5.6 we noted a simple example of a lower bound implying a BBH: the $\mathcal{C}$-BBH for $\mathcal{A}$ implies $\mathcal{C}$-SAT $\notin \mathcal{A}$. However, this lower bound is rather weak-looking: $\mathcal{C}$-SAT is NP-complete for many very simple $\mathcal{C}$. Are there circuit-analysis problems which are likely *not* to be NP-complete, which would still be implied by a Black-Box Hypothesis? We find this to be a very interesting question, and we currently do not have good candidates for such a problem.

5.7.2 Randomized Lower Bounds and Their Black-Box Hypotheses

We have shown that (deterministic) worst-case lower bounds can lead to results about analyzing circuits as boxes. What results can be derived from *average-case* or *randomized* lower bounds? We have obtained some preliminary results in this direction. For instance, if our analyst class $\mathcal{A}$ consists of *randomized* algorithms rather than deterministic ones, we can still prove connections between lower bounds against $\mathcal{A}$ and BBHs for $\mathcal{A}$, along the lines of Section 5.5. There are likely other connections like this to be found within the vast landscape of complexity theory.

5.7.3 Black-Box Hypotheses From More Lower Bounds?

While we have shown that various Black-Box Hypotheses do follow from certain lower bounds in a generic way, some lower bounds don't seem to imply a Black-Box Hypothesis. For example, a circuit-size hierarchy is well-known: for nice functions s, there are functions computable with size-$s(n)$ circuits that do not have circuits of size less than $s(n) - 5n$ (cf. [66]). This suggests the possibility that, for analysts $\mathcal{A}$ implemented by circuits of size less than $s(n) - 5n$, and boxes $\mathcal{C}$ which are circuits of size at least $s(n)$, the $\mathcal{C}$-Black-Box Hypothesis for $\mathcal{A}$ is true. However, our current methods are unable to prove such a sharp result. Are there other intermediate lower bounds (weaker than against e.g. $\mathcal{C}$-EVAL) that would still imply Black-Box Hypotheses?

Chapter 6

The Gotsman-Linial Conjecture is False

6.1 Polynomial Threshold Functions

In this chapter, we study a different complexity measure on Boolean functions. For the sake of simplicity, this chapter uses the set $\{-1,1\}^n$ to represent the Boolean hypercube, rather than the set $\{0,1\}^n$. All of the results carry over to the $\{0,1\}$ setting with a simple change of basis.

Definition 6.1.1 (Polynomial Threshold Function) *A function $f : \{-1,1\}^n \to \{-1,1\}$ is a* Polynomial Threshold Function of degree d *if it can be expressed as the sign of a polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ of degree at most d evaluated on the Boolean hypercube.*

Notation 6.1.1 (PTF) *For brevity, we will use the term (n,d)-PTF (or simply PTF, when n and d are either implicit or irrelevant) to refer to a polynomial threshold function of degree d on n variables.*

Definition 6.1.2 (Realizing Weights) *The coefficients of p are the* realizing weights *of f.*

Note that the realizing weights of a PTF are not unique, as any sufficiently small perturbation of p will not affect its sign on the discrete set $\{-1,1\}^n$. The concept of a PTF alone is not terribly exciting without restrictions on d, as every boolean function on n variables can be written as the sign of (and in fact can be written exactly as) a multilinear polynomial of degree n. We are interested particularly in the case where d is small.

In an influential paper, Craig Gotsman and Nathan Linial [53] applied Fourier analytic techniques to the study of PTFs. They were mainly interested in connecting different measures of the complexity of boolean functions, and of low-degree PTFs in particular. One such measure was the Average Sensitivity of a boolean function, defined in Fourier analytic terms. For simplicity, in this work we use the following (equivalent) combinatorial definition:

Definition 6.1.3 (Dichromatic Count) *For a function $f : \{-1,1\}^n \to \{-1,1\}$, we define its* dichromatic Count $\mathbf{D}[f]$ *to be the number of (unordered) pairs of Hamming neighbors $\{x, y\}$ such that $f(x) \neq f(y)$.*

We say that such a pair of Hamming neighbors is a *dichromatic edge* of f.

Definition 6.1.4 (Average Sensitivity) *The* Average Sensitivity *of a boolean function f is $\mathbf{AS}[f] := 2^{1-n}\mathbf{D}[f]$.*

Among other things, Gotsman and Linial proved a tight upper bound on the average sensitivity of $(n,1)$-PTFs, achieved by the MAJ function on n variables. They conjectured that this bound generalizes to higher degree PTFs, in that the (n,d)-PTF of maximal average sensitivity is the obvious symmetric candidate, which alternates signs on the $d+1$ values of $\sum_{i\in[n]} x_i$ closest to 0.

Conjecture 6.1.1 (Gotsman-Linial) *Let $p^*_{n,d}$ be the monic univariate polynomial of degree d with (non-repeated) roots at the d integers closest to 0 of opposite parity from n. Let $f^*(x_1,\ldots,x_n) = sgn\left(p^*_{n,d}\left(\sum_{i\in[n]} x_i\right)\right)$. Then for every (n,d)-PTF f, $\mathbf{AS}[f] \leq \mathbf{AS}[f^*_{n,d}]$.*

This conjecture was listed as a prominent open problem in [83] and [44]. If true, it would have many applications in complexity and learning (see for example [60, 52, 67, 69, 38]), although most of the applications would already be implied by an asymptotic version of the conjecture, stated below. Gotsman and Linial proved their conjecture for the case where $d = 1$, and it is also known to be true in the case where $d = 0$. However, it was left open whether the conjecture holds for any $d \geq 2$. Two weaker versions of this conjecture have since been formulated and studied.

Conjecture 6.1.2 (Gotsman-Linial - Asymptotic) *Let $f : \{-1, 1\}^n \to \{-1, 1\}$ be an (n, d)-PTF. Then the average sensitivity $\mathbf{AS}[f] \in O(d\sqrt{n})$.*

Conjecture 6.1.3 (Gotsman-Linial - Weak) *Let $f : \{-1, 1\}^n \to \{-1, 1\}$ be an (n, d)-PTF. Then the average sensitivity $\mathbf{AS}[f] \in O(\sqrt{n} \log^{g(d)} n)$ for some function g depending only on d.*

Conjecture 6.1.3 was resolved by Daniel Kane [68].

6.1.1 Result

In this chapter, we resolve the Gotsman-Linial Conjecture (Conjecture 6.1.1) for all pairs (n, d) except the case when $n > 7$ is even and $d = 2$. The main result of this chapter is the following.

Theorem 6.1.1 *For all pairs of natural numbers (n, d) satisfying one of the following criteria, there exists an (n, d)-PTF $f_{n,d}$ witnessing a counterexample to the Gotsman-Linial Conjecture (Conjecture 6.1.1):*

- *$n \geq 5$ is odd, and $d = 2$.*
- *$n \geq 7$, and $3 \leq d \leq n - 3$.*

Moreover, $\mathbf{AS}[f_{n,d}] \in (1 + \Omega(n^{-1}e^{-d^2/n}))\mathbf{AS}[f^*_{n,d}]$.

In addition, the conjecture holds in many of the remaining cases.

Theorem 6.1.2 *For all pairs of natural numbers (n, d) satisfying one of the following criteria, $f^*_{n,d}$ has the greatest average sensitivity among (n, d)-PTFs.*

- $d \leq 1$.
- $d \geq n - 2$.
- $n = 6$.

Our results (and the remaining open cases) are summarized in Figure 6-1. Although we refute the Gotsman-Linial Conjecture for most cases that are of interest for applications, the asymptotic conjecture (Conjecture 6.1.2), which would suffice for most known applications, remains open.

n \ d	0	1	2	3	4	5	6	7	8	$\cdots$
1	✓	✓								
2	✓	✓	✓							
3	✓	✓	✓	✓						
4	✓	✓	✓	✓	✓					
5	✓	✓	×	✓	✓	✓				
6	✓	✓	✓	✓	✓	✓	✓			
7	✓	✓	×	×	×	✓	✓	✓		
8	✓	✓	?	×	×	×	✓	✓	✓	
9	✓	✓	×	×	×	×	×	✓	✓	$\ddots$
10	✓	✓	?	×	×	×	×	×	✓	$\ddots$
11	✓	✓	×	×	×	×	×	×	×	$\ddots$
12	✓	✓	?	×	×	×	×	×	×	$\ddots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
$2k$	✓	✓	?	×	×	×	×	×	×	$\ddots$
$2k+1$	✓	✓	×	×	×	×	×	×	×	$\ddots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Figure 6-1: Results are summarized in the above table. A cyan tick mark indicates a case in which the conjecture holds (for all (n, d)-PTFs f, $\mathbf{AS}[f] \leq \mathbf{AS}[f^*_{n,d}]$). A red cross indicates a refutation (there exists an (n, d)-PTF f such that $\mathbf{AS}[f] \in (1 + \Omega(n^{-1}e^{-d^2/n}))\mathbf{AS}[f^*_{n,d}]$). A black question mark indicates an open case. Note: the cases $(n, d) = (6, 2)$ and $(n, d) = (6, 3)$ were verified with the help of a computer search and a linear program solver (see Section 6.5).

The remainder of this chapter is structured as follows. We first present some high

level intuition relating to the Gotsman-Linial Conjecture. Section 6.2 contains background information. Section 6.3 contains constructions of the refutations indicated in Figure 6-1. Section 6.4 concludes and presents a revised conjecture. Section 6.5 contains details on how a computer search aided the proofs.

6.1.2 Intuition

We start with some very high level intuition as to why the Gotsman-Linial Conjecture might be (approximately) true. The conjecture holds in the case of symmetric PTFs (boolean functions which can be expressed as the sign of a univariate polynomial in the sum of the input bits). This follows from the Fundamental Theorem of Algebra and a simple counting argument. In the more general case, we might expect that a degree-d PTF can be expressed (at least approximately) in terms of d unate functions. This generalizes the observation that every linear threshold function is unate. For a sufficiently close approximation, this would prove the Asymptotic Gotsman-Linial Conjecture. Intuition may also be drawn from Kane's proof of Conjecture 6.1.3. If inputs are chosen from a Gaussian distribution instead of a Bernoulli distribution, a polynomial p is expected to be too large in magnitude for a small change in its input to change its sign. Under certain conditions, a similar result can be extended to polynomial threshold functions on the Boolean hypercube.

As for why the Gotsman-Linial Conjecture is not (exactly) true, we observe that the PTF of conjectured maximal average sensitivity is the product of d linear threshold functions, with parallel separating hyperplanes between two of the middle $d+1$ layers (sets of vertices of equal Hamming weight) in the hypercube. For some d, one might expect to be able to find a PTF of greater average sensitivity approximated by turning one of these separating hyperplanes "sideways", i.e. replacing a hyperplane that cuts the fewest edges with a hyperplane orthogonal to the rest. Intuitively, this would require that d be sufficiently large that some of the hyperplanes cut many more edges than others, but also sufficiently small that not too many edges are cut by two hyperplanes. As it turns out, this intuition can be formalized for many n and d,

refuting the Gotsman-Linial Conjecture.

6.2 Preliminaries

6.2.1 Background

Low-degree PTFs, in particular degree-1 PTFs with integral and polynomially bounded realizing weights, are of interest in the study of complexity classes such as TC and of neural networks.

Definition 6.2.1 (Polynomial Threshold Circuit) *A circuit (with unbounded fan-in) is a* degree-d polynomial threshold circuit *if each of its constituent gates computes a degree-d PTF of its inputs.*

Definition 6.2.2 (Linear Threshold Circuit) *A* linear threshold circuit *(resp. linear threshold function) is a degree-*1 *polynomial threshold circuit (resp. PTF).*

Recall alternating (AC) and threshold (TC) circuits from Section 2.2.

Example 6.2.1 *Since* And, Or, Maj, *and* Not *are all linear threshold functions,* AC *and* TC *circuits are linear threshold circuits.*

Despite much research, the power of polynomial threshold circuits is poorly understood. For instance, it is currently an open question, and a rather embarrassing one at that, whether NE (the class of functions computable in nondeterministic $2^{O(n)}$ time) is contained in TC^0_3 (the class of functions computable with depth-three, polynomial size TC circuits). Recent work by Daniel Kane and Ryan Williams [69] gave a partial answer to this question. They studied the sensitivity of PTFs to random restrictions, proving (among other things) that NE (and in fact, P-uniform TC^0) does not have depth-3 TC circuits of $n^{1.499}$ gates or $n^{2.499}$ wires.

6.2.2 Progress

Conjecture 6.1.1 is trivially true in the cases $d = 0$ and $d = n$ (the only $(n, 0)$-PTFs are the constant functions, and $f^*_{n,n}$ is the parity function, which has the maximum possible average sensitivity). Gotsman and Linial originally noted that Conjecture 6.1.1 had already been proven in the case where $d = 1$ by Patrick O'Neil in 1971 [86].

Theorem 6.2.1 (O'Neil) *The maximal number k of edges of $H := \{-1, 1\}^n$ which may be cut by a hyperplane P is given by $k = \left\lceil \frac{n}{2} \right\rceil \binom{n}{\lfloor n/2 \rfloor}$.*

Very little additional progress was made towards resolving the above conjectures until recently. The first non-trivial bounds on the average sensitivity of PTFs of arbitrary degree were found independently by two groups [60, 43] and published jointly [42]. Daniel Kane in 2012 obtained the first bound which was truly sublinear in n [67], and in 2013, he proved the weak version of the Gotsman-Linial Conjecture (Conjecture 6.1.3) [68].

6.3 Resolution of Gotsman-Linial Conjecture

For readability, we start by introducing some notation.

Definition 6.3.1 *Let $f, g : \{-1, 1\}^n \to \{-1, 1\}$. We say $f \sim g$, iff there exist $\sigma \in S_n$ and $\alpha \in \{-1, 1\}^n$ such that the function $x \mapsto f(x_1, \ldots, x_n)\, g(\alpha_1 x_{1\sigma}, \ldots, \alpha_n x_{n\sigma})$ is a constant.*

Note that $\sim$ defines an equivalence relation on boolean functions. Two functions are equivalent iff one can be turned into the other through a combination of permuting the inputs and negating the inputs/output.

Definition 6.3.2 (Hypersensitivity) *An (n, d)-Hypersensitive Function, or (n, d)-*HSF *is an (n, d)-PTF f such that $\mathbf{D}[f] > \mathbf{D}[f^*_{n,d}]$.*

More generally, we say that a PTF f is an HSF if n and d are either implicit or irrelevant. We may now restate the original Gotsman-Linial Conjecture (Conjecture 6.1.1) as follows:

Conjecture 6.3.1 *For all $n, d \in \mathbb{N}$, (n, d)-HSFs do not exist.*

We first prove some simple cases of Conjecture 6.3.1. The following corollary of O'Neil's theorem (Theorem 6.2.1) uses our notation.

Corollary 6.3.1 *For every $n \in \mathbb{N}$, $(n, 1)$-HSFs do not exist.*

Proof. Every $(n, 1)$-PTF f is defined by a separating hyperplane P which cuts all of the dichromatic edges of f. From O'Neil, $\mathbf{D}[f] \leq \left\lceil \frac{n}{2} \right\rceil \binom{n}{\lfloor \frac{n}{2} \rfloor} = \mathbf{D}[f^*_{n,1}]$, so f is not an HSF. □

The case $d = n-1$ is a simple consequence of a result first proven in 1968 by Marvin Minsky and Seymour Papert [76] and since re-proven several times. We present here a variation on the proof by Aspnes *et al.* [11].

Theorem 6.3.1 (Minsky-Papert) *Any PTF which computes parity on n variables must have degree at least n.*

Proof. Let $p \in \mathbb{R}[x_1, \ldots, x_n]$ be a multilinear polynomial of degree $n-1$ which is never zero on $\{-1, 1\}^n$. The set of monomials of degree at most n is an orthogonal basis for the vector space of degree-n multilinear polynomials on the boolean hypercube. Hence p is orthogonal to the parity function ϕ_n, i.e. $\langle p, \phi_n \rangle = \sum_{x \in \{-1,1\}^n} p(x)\phi_n(x) = 0$. By assumption, every term in the sum on the RHS is non-zero, so at least one of them is negative, i.e. $\operatorname{sgn} \circ p \neq \phi_n$. □

Corollary 6.3.2 *For every $n \in \mathbb{N}$, $(n, n-1)$-HSFs do not exist.*

Proof. Let f be an $(n, n-1)$-PTF. Then $f \neq \phi_n$, and $f \neq -\phi_n$. Let $X = \{x : f(x) = \phi_n(x)\}$ and $Y = \{y : f(y) \neq \phi_n(y)\}$. Take $x \in X$ and $y \in Y$. There are n edge-disjoint paths between x and y in the boolean hypercube, and each must contain at least one edge crossing the cut between X and Y (i.e. a monochromatic edge). Hence $\mathbf{D}[f] \leq n(2^{n-1} - 1) = \mathbf{D}[f^*_{n,n-1}]$, so f is not an HSF. □

Lemma 6.3.1 *Let $n, d \in \mathbb{N}$, and let g have maximal $\mathbf{D}[g]$ over all $(n-1, d)$-PTFs. Then for every (n, d)-PTF f, $\mathbf{D}[f] \leq \frac{2n}{n-1}\mathbf{D}[g]$.*

Proof. Let $n, d \in \mathbb{N}$, and let g be an $(n-1, d)$-PTF with $\mathbf{D}[g]$ maximal. Let f be an (n, d)-PTF. Any restriction f' of f to a function on $n-1$ variables is also a degree-d PTF, so $\mathbf{D}[f'] \leq \mathbf{D}[g]$. There are $2n$ such restrictions f', and each dichromatic edge of f appears in exactly $n-1$ of them. Hence $(n-1)\mathbf{D}[f] \leq 2n\mathbf{D}[g]$, from which the desired result follows immediately. □

Lemma 6.3.2 *Let $n, d \in \mathbb{N}$ with $d < n$. If n and d have the same parity, and $(n-1, d)$-HSFs do not exist, then (n, d)-HSFs do not exist.*

Proof. Assume that no $(n-1, d)$-PTF is an HSF. If n and d have the same parity, then every restriction of $f^*_{n,d}$ is equivalent (with respect to $\sim$) to $f^*_{n-1,d}$. There are $2n$ such restrictions, and each dichromatic edge of $f^*_{n,d}$ appears in exactly $n-1$ of them, so $\mathbf{D}[f^*_{n,d}] = \frac{2n}{n-1}\mathbf{D}[f^*_{n-1,d}]$. Hence by Lemma 6.3.1, (n, d)-HSFs do not exist. □

Corollary 6.3.3 *For every $n \in \mathbb{N}$, $(n, n-2)$-HSFs do not exist.*

Proof. This follows from Corollary 6.3.2 and Lemma 6.3.2. □

Corollary 6.3.4 *Let $n, d \in \mathbb{N}$. If $d \leq n \leq 5$ and $(n, d) \neq (5, 2)$, then (n, d)-HSFs do not exist.*

Proof. This follows from Corollaries 6.3.1, 6.3.2 and 6.3.3, and the fact that (n, d)-HSFs trivially do not exist when $d \in \{0, n\}$. □

6.3.1 A Simple Counterexample

In the statement of Corollary 6.3.4, the caveat $(n, d) \neq (5, 2)$ cannot be removed.

Lemma 6.3.3 *There exists a unique $(5, 2)$-HSF $f_{5,2}$, modulo $\sim$.*

Proof. In the case where $n = 5$ and $d = 2$, $p^*_{5,2}(x) = x(x-2)$, and $\mathbf{D}[f^*_{5,2}] = 50$. Let $q \in \mathbb{R}[x, y]$ be defined by $q(x, y) := 3y^2 - x^2 + 2xy + y - x - 3$, let $q' \in \mathbb{R}[x_1, \ldots, x_5]$ such that $q'(x_1, \ldots, x_5) := q(x_1 + x_2, x_3 + x_4 + x_5)$, and let $f_{5,2} := \operatorname{sgn} \circ q'$. Since q

is quadratic, $f_{5,2}$ is a $(5,2)$-PTF. It is not difficult to verify that $\mathbf{D}[f_{5,2}] = 51 > 50 = \mathbf{D}[f^*_{5,2}]$, so $f_{5,2}$ is a $(5,2)$-HSF. For uniqueness, we performed an exhaustive search using linear programming. For details, see Section 6.5.1. □

The existence of a $(5,2)$-HSF precludes the use of Lemma 6.3.2 to prove that $(6,2)$-HSFs do not exist. However, the uniqueness of $f_{5,2}$, along with the fact that it only has one additional dichromatic edge, allows for a proof using Lemma 6.3.1.

Lemma 6.3.4 *For every d, $(6,d)$-HSFs do not exist.*

Proof. The cases $d \in \{0,1,4,5,6\}$ have already been covered. For $d = 3$, see Section 6.5.3. The case $d = 2$ remains. Assume for the sake of contradiction that f is a $(6,2)$-HSF. The dichromatic count of every boolean function on an even number of variables is an even integer. Since for every $(5,2)$-PTF g, $\mathbf{D}[g] \leq 51$, Lemma 6.3.1 implies that $120 < \mathbf{D}[f] \leq 122.4$, and hence that $\mathbf{D}[f] = 122$. There are 12 restrictions of f to a function g on 5 variables, all of which satisfy $\mathbf{D}[g] \leq 51$. Every dichromatic edge in f appears in exactly five such g, so the expectation over a uniformly random restriction g of $\mathbf{D}[g]$ is $\frac{5}{12} \cdot 122 > 50.5$. Since $\mathbf{D}[g]$ is always an integer, $\mathbf{D}[g] = 51$ with probability strictly greater than $1/2$. In particular, there exists i such that $f|_{x_i=-1} \sim f|_{x_i=1} \sim f_{5,2}$ (*). However, it is easily verified (see Section 6.5.2) that no function f satisfying both (*) and $\mathbf{D}[f] = 122$ is a $(6,2)$-PTF. This contradicts the initial choice of f. Hence no $(6,2)$-HSFs exist. □

This also completes the proof of Theorem 6.1.2. □

6.3.2 Extension to Odd n

We may extend $f_{5,2}$ to an $(n,2)$-HSF for any odd $n \geq 5$.

Theorem 6.3.2 *For every odd $n \in \mathbb{N}$ with $n \geq 5$, there exists an $(n,2)$-HSF $f_{n,2}$ with*

$\mathbf{D}[f_{n,2}] \in \left(1 + \Omega\left(n^{-1}\right)\right) \mathbf{D}[f^*_{n,2}]$.

Intuitively, $f_{n,2}$ behaves exactly as $f_{5,2}$, with the additional variables contributing to the second argument of q.

Proof. Let $n \geq 5$ be an odd integer. Let $A := \{-2, 0, 2\}$ and $B := 2\mathbb{Z} + 1$. Let H be the n-dimensional boolean hypercube, and let G be the graph with vertex set $A \times B$ and an edge between u and v exactly when $\|u - v\|_1 = 2$. Let $\phi : H \to G$ be the graph homomorphism defined by $\phi(x_1, \ldots, x_n) := (x_1 + x_2, x_3 + \ldots + x_n)$. Let $q(x, y) := 3y^2 - x^2 + 2xy + y - x - 3$ as above, let $f := \operatorname{sgn} \circ q$, and take $f_{n,2} := f \circ \phi$. Note that because ϕ is a graph homomorphism, we may compute $\mathbf{D}[f_{n,2}]$ by counting the dichromatic edges e induced by f on G, weighted by $\phi^{-1}(e)$. To this end, we observe that an edge e between $(2i-2, 2j+2-n)$ and $(2i-2, 2j-n)$ has a preimage under ϕ of cardinality

$$|\phi^{-1}(e)| = \binom{2}{i}\binom{n-2}{j}(n-2-j)\,.$$

Similarly, for an edge e between $(2i-2, 2j+2-n)$ and $(2i, 2j+2-n)$,

$$|\phi^{-1}(e)| = \binom{2}{i}\binom{n-2}{j}(2-i)\,.$$

We observe that q is positive on $A \times B$ except at the four points $\{(-2, 1), (0, -1), (2, -1), (2, 1)\}$. Hence f gives nine dichromatic edges, as indicated by the black lines in Figure 6.3.2.

		j							
		$2-n$	$\cdots$	-3	-1	$+1$	$+3$	$\cdots$	$n-2$
	-2	$+$	$\cdots$	$+$	$+$	$-$	$+$	$\cdots$	$+$
i	0	$+$	$\cdots$	$+$	$-$	$+$	$+$	$\cdots$	$+$
	$+2$	$+$	$\cdots$	$+$	$-$	$-$	$+$	$\cdots$	$+$

Figure 6-2: Illustration of $(n, 2)$-HSF

Summing the above expressions over these nine edges, we have

$$\begin{aligned}
\mathbf{D}[f_{n,2}] &= \binom{n-2}{\frac{n-1}{2}}\left(\binom{2}{0}n + \binom{2}{1}(n-1) + \binom{2}{2}(n-1)\right) \\
&= \binom{n-2}{\frac{n-1}{2}}(4n-3) \\
&= \binom{n-2}{\frac{n-1}{2}}(n-3+3n) \\
&= \binom{n-2}{\frac{n+1}{2}}(n+1) + \binom{n-2}{\frac{n-1}{2}}3n \\
&= \left(\binom{n-1}{\frac{n+1}{2}} + \binom{n-1}{\frac{n-1}{2}}\right)n + \binom{n-2}{\frac{n+1}{2}} \\
&= \binom{n}{\frac{n+1}{2}}n + \binom{n-2}{\frac{n+1}{2}} \\
&\in \left(1+\Theta(n^{-1})\right)\mathbf{D}[f^*_{n,2}].
\end{aligned}$$

Hence $f_{n,2}$ is an $(n,2)$-HSF, as desired. □

6.3.3 The General Case

Using a similar construction, we now prove the existence of HSFs of arbitrary degree.

Theorem 6.3.3 *For every $n, d \in \mathbb{N}$ with $n \geq 7$ and $3 \leq d \leq n-3$, there exists an (n,d)-HSF $f_{n,d}$ with $\mathbf{D}[f_{n,d}] \in \left(1+\Omega\left(n^{-1}e^{-d^2/n}\right)\right)\mathbf{D}[f^*_{n,d}]$.*

We first consider the case where n and d have the same parity. The case where n and d have opposite parity is similar but handled later.

Theorem 6.3.4 *For every $n, d \in \mathbb{N}$ with $3 \leq d \leq n-4$ and $n-d$ even, there exists an (n,d)-HSF $f_{n,d}$ with $\mathbf{D}[f_{n,d}] \in \left(1+\Omega\left(n^{-1}e^{-d^2/n}\right)\right)\mathbf{D}[f^*_{n,d}]$.*

Proof. Let n, d be integers of the same parity with $3 \leq d \leq n-4$. Let $A := \{-3,-1,1,3\}$ and let $B := 2\mathbb{Z}+d+1$. Let H be the n-dimensional boolean hypercube, and let G be the graph with vertex set $A \times B$ and an edge between u and v exactly when $\|u-v\|_1 = 2$. Let ϕ be the graph homomorphism defined by $\phi(x_1, \ldots x_n) :=$

$(x_1 + x_2 + x_3, x_4 + \ldots + x_n)$. We now define four polynomials p_1, p_2, p_3, p_4 on $A \times B$ as follows:

$$\begin{aligned}
p_1(x,y) &:= (y - 1 + d)(y + 1 - d) \\
p_2(x,y) &:= 1 - 2\left(x(d-1) + y\right)^2 \\
p_3(x,y) &:= (y - 3 + d)(y - 5 + d) \cdots (y + 5 - d)(y + 3 - d) \\
p_4(x,y) &:= x(x+2)(x-2)(y - 4 + d)(y - 6 + d) \cdots (y + 6 - d)(y + 4 - d)
\end{aligned}$$

Since $p_1 \in \Omega(p_2)$, there exists $\varepsilon' > 0$ such that for every $v \in G$ with $p_1(v) \neq 0$, $|p_1(v)| > |2\varepsilon' p_2(v)|$. Similarly, $p_3 \in \Omega(p_4)$, so there exists $\varepsilon \in (0, \varepsilon']$ such that for every $v \in G$ with $p_3(v) \neq 0$, $|p_3(v)| > |\varepsilon p_4(v)|$. For instance, we may take $\varepsilon = \varepsilon' = (4d)^{-d}$. Take $p := (p_1 + \varepsilon p_2) \cdot p_3 - \varepsilon^2 p_4$, take $g := \operatorname{sgn} \circ p$, and take $f_{n,d} := g \circ \phi$. Since p_1 and p_2 have degree 2, p_3 has degree $d-2$, and p_4 has degree d, $f_{n,d}$ is a polynomial threshold function of degree d. Towards computing $\mathbf{D}[f_{n,d}]$, we first consider the relevant behaviors of p_1, p_2, p_3, p_4 separately. All four are integer-valued (evaluations of) polynomials on the domain $A \times B$. Both p_2 and p_4 are always odd, so in particular, are non-zero everywhere. Firstly, p_3 is positive when $y > d-3$, is zero when $|y| \leq d-3$, and has the same sign as $(-1)^d$ when $y < 3-d$. Clearly, p_1 is positive when $|y| > d-1$ and zero when $|y| = d-1$. By choice of ε, $p_1 + \varepsilon p_2$ is never in the interval $(-\varepsilon, \varepsilon)$ and always has the same sign as p_1 when p_1 is non-zero. Similarly, p is always non-zero and always has the same sign as $(p_1 + \varepsilon p_2) \cdot p_3$ when p_3 is non-zero. Hence we may rewrite g as the following piecewise function:

$$g(x,y) = \begin{cases}
(-1)^d & y < 1 - d \\
\operatorname{sgn}((-1)^d p_2(x,y)) & y = 1 - d \\
\operatorname{sgn}(p_4(x,y)) & |y| < d - 1 \\
\operatorname{sgn}(p_2(x,y)) & y = d - 1 \\
1 & y > d - 1
\end{cases}$$

Since p_2 is positive only at the two points $(-1, d-1)$ and $(1, 1-d)$ when $|y| = d-1$, the above piecewise representation shows that when $|y| \leq d-1$, $g(x,y)$ computes the

parity function except at the two points $(3, d-1)$ and $(-3, 1-d)$ (illustrated in Figures 6-3 and 6-4). We now define $g' : A \times B \to \{-1, 1\}$ by $g'(\phi_{n,3}(x)) := -f^*_{n,d}(x)$. Note

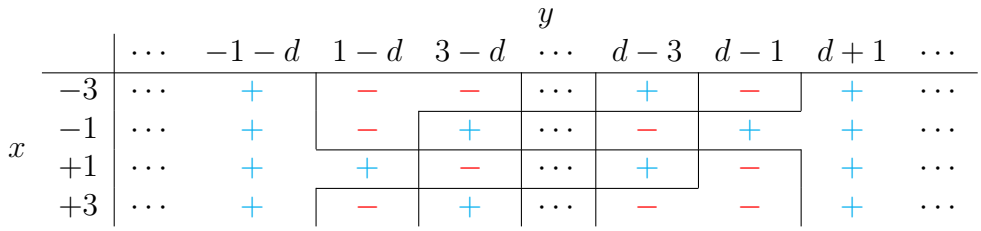


Figure 6-3: Illustration of g in the case where n and d are both even

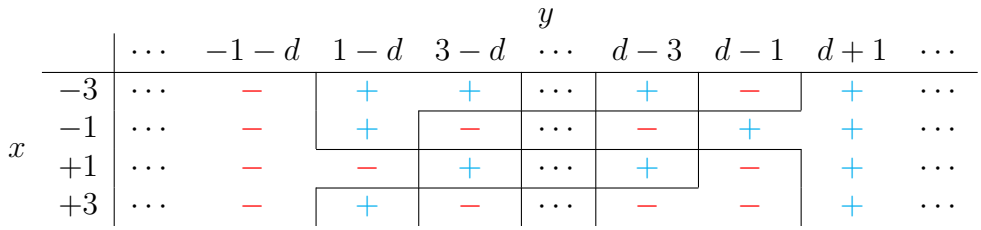


Figure 6-4: Illustration of g in the case where n and d are both odd

that because $f^*_{n,d}$ is symmetric, this gives a well-defined function g'. It is easily verified that for all $(x, y) \in A \times B$ such that $|y| \leq d-1$ and at the two points $(3, -1-d)$ and $(-3, d+1)$, $g'(x, y) = g(x, y)$, and that for all other $(x, y) \in A \times B$, $g'(x, y) = -g(x, y)$. Hence there are ten edges $\{u, v\}$ in G for which $g(u)g(v) \neq g'(u)g'(v)$. This allows us

to compute $\mathbf{D}[f_{n,d}]$ as follows:

$$\begin{aligned}
\mathbf{D}[f_{n,d}] &= \mathbf{D}[f^*_{n,d}] \\
&\quad + (n+d-2)\binom{n-3}{\frac{n-d-4}{2}} + 3(n+d-2)\binom{n-3}{\frac{n-d-4}{2}} \\
&\quad - 3(n+d-2)\binom{n-3}{\frac{n-d-4}{2}} - 6\binom{n-3}{\frac{n-d-4}{2}} - (n-d-4)\binom{n-3}{\frac{n-d-4}{2}} \\
&= \mathbf{D}[f^*_{n,d}] + (2d-4)\binom{n-3}{\frac{n-d-4}{2}} \\
&\in \left(1 + \Omega\left(\frac{\binom{n}{\frac{n-d}{2}}}{n\binom{n}{\frac{n}{2}}}\right)\right)\mathbf{D}[f^*_{n,d}] \\
&\subseteq \left(1 + \Omega\left(n^{-1}e^{-d^2/n}\right)\right)\mathbf{D}[f^*_{n,d}]. && \text{(Stirling's Inequality)}
\end{aligned}$$

Hence $f_{n,d}$ is an (n,d)-HSF, as desired. □

Theorem 6.3.5 *For every $n, d \in \mathbb{N}$ with $n \geq 7$, $3 \leq d \leq n-3$ and $n-d$ odd, there exists an (n,d)-HSF $f_{n,d}$ with $\mathbf{D}[f_{n,d}] \in \left(1 + \Omega\left(n^{-1}e^{-d^2/n}\right)\right)\mathbf{D}[f^*_{n,d}]$.*

Proof. The proof proceeds similarly to the previous case. We define A, B, H, G, p_1, p_2, p_3, p_4, p, g, and g' as above, and we define $\psi(x_1, \ldots, x_n) := x_1 + x_2 + x_3, 1 + x_4 + \ldots + x_n)$. We now define $f_{n,d} := g \circ \psi$ analogously to above. The computation

of $\mathbf{D}[f_{n,d}]$ now proceeds as follows:

$$\begin{aligned}
\mathbf{D}[f_{n,d}] &= \mathbf{D}[f^*_{n,d}] \\
&\quad + \frac{n+d-3}{2}\binom{n-3}{\frac{n-d-3}{2}} + 3\frac{n+d-3}{2}\binom{n-3}{\frac{n-d-3}{2}} \\
&\quad - 3\frac{n+d-3}{2}\binom{n-3}{\frac{n-d-3}{2}} - 3\binom{n-3}{\frac{n-d-3}{2}} - \frac{n-d-3}{2}\binom{n-3}{\frac{n-d-3}{2}} \\
&\quad + \frac{n+d-1}{2}\binom{n-3}{\frac{n-d-5}{2}} + 3\frac{n+d-1}{2}\binom{n-3}{\frac{n-d-5}{2}} \\
&\quad - 3\frac{n+d-1}{2}\binom{n-3}{\frac{n-d-5}{2}} - 3\binom{n-3}{\frac{n-d-5}{2}} - \frac{n-d-5}{2}\binom{n-3}{\frac{n-d-5}{2}} \\
&= \mathbf{D}[f^*_{n,d}] + (d-3)\binom{n-3}{\frac{n-d-3}{2}} + (d-1)\binom{n-3}{\frac{n-d-5}{2}} \\
&\in \left(1 + \Omega\left(\frac{\binom{n}{\frac{n-d}{2}}}{n\binom{n}{\frac{n}{2}}}\right)\right)\mathbf{D}[f^*_{n,d}] \\
&\subseteq \left(1 + \Omega\left(n^{-1}e^{-d^2/n}\right)\right)\mathbf{D}[f^*_{n,d}]. && \text{(Stirling's Inequality)}
\end{aligned}$$

Hence $f_{n,d}$ is an (n,d)-HSF, as desired. □

This also completes the proofs of Theorems 6.3.3 and 6.1.1. □

6.4 Conclusion

For almost all d and almost all n, we refute the Gotsman-Linial Conjecture (Conjecture 6.1.1) with a multiplicative separation of $1 + \Theta_d\left(n^{-1}\right)$. This separation is too weak to refute most known applications of the conjecture. We would need to improve $1+\Theta_d\left(n^{-1}\right)$ to $\omega(1)$ to refute the Asymptotic Gotsman-Linial Conjecture (Conjecture 6.1.2), on which the applications depend. Although for every (n,d)-HSF f given in this chapter, $\mathbf{D}[f] > \mathbf{D}[f^*_{n,d}]$, it should be noted that the RHS is still an upper bound in a *limiting* sense. This, along with the intuition presented in Section 6.1.2, invites the following revised conjecture.

Conjecture 6.4.1 (Gotsman-Linial - Limit) *Let $f : \{-1,1\}^n \to \{-1,1\}$ be an*

(n, d)-*PTF. Then the average sensitivity* $\mathbf{AS}[f] \leq d\mathbf{AS}[f^*_{n,1}]$.

Conjecture 6.4.1 would resolve the remaining cases of Conjectures 6.3.1 and 6.1.1, i.e.

Conjecture 6.4.2 *For every even* n, $(n, 2)$*-HSFs do not exist.*

Furthermore, our revised conjecture would imply the Asymptotic Gotsman-Linial Conjecture (Conjecture 6.1.2) and its consequent applications.

6.5 Search with Linear Programming

Here we describe how a computer search resolved the cases of $n = 6$ and $d = 2, 3$ of the Gotsman-Linial Conjecture. First, we note that the problem of determining whether a boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$ is an (n, d)-PTF is equivalent to determining whether a particular linear program has any feasible solution. Unfortunately, leveraging this fact to compute the maximal average sensitivity of an (n, d)-PTF with a naïve exhaustive search takes doubly exponential time so is intractable for large n (i.e. $n > 4$). However, by using Lemma 6.3.1, we can conduct a more efficient search. We maintain a partial function f and conduct a DFS in which we define f successively on inputs in increasing order of Hamming weight. This allows us to keep bounds on $\mathbf{D}[f]$ by counting the edges that are already constrained to be monochromatic or dichromatic. When $\mathbf{D}[f]$ becomes too low or too high, we can prune the search and backtrack before fully defining f, allowing (tolerably) efficient searches up to $n = 6$.

6.5.1 $(n, d) = (5, 2)$

In the case $(n, d) = (5, 2)$, Lemma 6.3.1 implies that for every $(5, 2)$-HSF f, we have $50 < \mathbf{D}[f] \leq 60$. Because a random boolean function on 5 variables has far fewer than 50 dichromatic edges with high probability, most search branches are pruned early. The modified search confirmed that every $(5, 2)$-HSF f with $50 < \mathbf{D}[f] \leq 60$ satisfies $f \sim f_{5,2}$, and hence that $f_{5,2}$ is the unique $(5, 2)$-HSF.

6.5.2 $(n, d) = (6, 2)$

The proof of Lemma 6.3.4 relies on the claim that for every $(6, 2)$-PTF f and for every variable x_i, either $f|_{x_i=-1} \not\sim f_{5,2}$, $f|_{x_i=1} \not\sim f_{5,2}$, or $\mathbf{D}[f] \neq 122$. By symmetry, we may remove the dependence on i, and we may turn one of the equivalences into an equality. The above is equivalent to the claim that for every $(6, 2)$-PTF f, either $f|_{x_1=-1} \not\sim f_{5,2}$, $f|_{x_1=1} \neq f_{5,2}$, or $\mathbf{D}[f] \neq 122$. Because a restriction of a $(6, 2)$-PTF is a $(5, 2)$-PTF and must therefore have no more than 51 dichromatic edges, it suffices to show that for every function $f : \{-1, 1\}^6 \to \{-1, 1\}$ such that $f|_{x_1=-1} \sim f|_{x_1=1} = f_{5,2}$ and $\mathbf{D}[f] = 122$, there exist i and b such that $\mathbf{D}[f|_{x_i=b}] > 51$. There are fewer than 1000 functions f satisfying both $f|_{x_1=-1} \sim f|_{x_1=1} = f_{5,2}$ and $\mathbf{D}[f] = 122$ (and these are easily enumerated), and each has only 10 relevant restrictions to five variables, so the last claim is easily verified with a quick computer search.

6.5.3 $(n, d) = (6, 3)$

In the case $(n, d) = (6, 3)$, Lemma 6.3.1 implies that for every $(6, 3)$-HSF f, we have $150 < \mathbf{D}[f] \leq 168$. Because a random boolean function on 6 variables has far fewer than 150 dichromatic edges with overwhelming probability, most search branches are pruned early. The modified search confirmed that no $(6, 3)$-PTF f satisfying $150 < \mathbf{D}[f] \leq 168$ is an HSF, and hence that no $(6, 3)$-HSFs exist.

Chapter 7

Smaller Counting Circuits for Symmetric Functions

7.1 Counting Circuits

This chapter studies the complexity class CC^0 and related classes. Recall from Definition 2.2.17 that CC^0 consists of constant-depth circuits in which every (unbounded fan-in) gate (called a $\textsc{Mod}_m$ gate) determines whether the sum of its inputs is divisible by a small constant integer m. Although the model looks rather peculiar, CC^0 circuits (a.k.a. pure-ACC circuits [105]) have been a longstanding and fundamental roadblock in the way of improved circuit complexity lower bounds. Since their identification over 30 years ago [15, 19], scant progress has been made on lower bounds against CC^0 circuits, and their close cousin ACC^0 which includes And and Or in the gate basis. Some exceptions include work focusing on special cases of the problem (e.g., [16, 55, 35, 36]), uniform lower bounds [3], and work proving strong lower bounds but only for functions whose complexity is in $\mathsf{QuasiNP}$ or higher (e.g., [103, 39, 78, 37]). If there has ever been a "circuit complexity winter", CC^0 circuits are at least partly to blame.

Besides our own ignorance, could there be deeper reasons why CC^0 circuits have been so difficult for showing limitations? Here we explore the possibility that CC^0 circuits may be powerful, focusing on the natural class of *symmetric Boolean functions*

whose output depends only on the number of ones in the input. Although it has been conjectured for many years that the AND function does not have polynomial-size CC^0 circuits ([17, 100, 99])[1] our results show that low-depth MOD_m circuits with arbitrary but fixed modulus m can actually compute arbitrary symmetric Boolean functions (such as MAJ) much more efficiently than low-depth circuits with AND, OR, and MOD_q gates, when q is a prime power.

It is well-known that AC^0 circuits, which consist of AND, OR, NOT gates and have constant-depth, require $\exp(\Omega(n^{1/(d-1)}))$ size to compute arbitrary symmetric functions in depth d [61]. In recent work, Oliveira, Santhanam, and Srinivasan [85] have shown that PARITY gates (a.k.a. MOD_2 or $\oplus$ gates) can help compute symmetric functions more efficiently than what AND, OR, NOT can accomplish in constant depth. In particular, they show that $\mathsf{AC}^0[2]$ circuits (with AND, OR, and PARITY) of depth 4 can compute MAJ in $\exp(\Theta(n^{1/4}))$ size, depth $d \geq 5$ can compute symmetric functions in size $\exp(\widetilde{O}(n^{\frac{2}{3(d-4)}}))$, and they show a size lower bound of $\exp(\Omega(n^{1/(2d-4)}))$ for the MAJ function, improving [90, 98].

Smaller MOD_m Circuits. Could even smaller circuits for symmetric functions be achieved using MOD_m gates, for other composite m? It turns out that this is possible. In fact, even in depth three, any symmetric function can computed with a MOD_m circuit of size 2^{n^ε} for any desired $\varepsilon > 0$.

Theorem 7.1.1 *For every $\varepsilon > 0$, there is a modulus $m \leq (1/\varepsilon)^{2/\varepsilon}$ such that every symmetric function on n bits can be computed by depth-3 MOD_m circuits of $\exp(O(n^\varepsilon))$ size. In fact, the circuits have the form $\text{MOD}_{p_1} \circ \text{MOD}_{p_2 \cdots p_r} \circ \text{MOD}_{p_1}$, where $p_1, \ldots, p_r$ are distinct primes.*[2]

That is, without any AND/OR gates, we can obtain CC^0 circuits with a substantially smaller number of gates than the longstanding lower bounds for $\mathsf{AC}^0[q]$ circuits

[1] Hansen and Koucký [58] give an interesting counterpoint, showing that *probabilistic* CC^0 circuits can compute AND efficiently. Thus the AND $\in \mathsf{CC}^0$ problem is equivalent to a derandomization question.

[2] The $G \circ H \circ I$ notation means that the output gate has type G, on the middle layer there are gates only of type H, and on the bottom layer (nearest the inputs) there are only gates of type I.

computing symmetric functions (mentioned two paragraphs ago), for prime power q.

It has been known for decades [18] that *depth*-2 $\text{MOD}_p \circ \text{MOD}_m$ circuits (and $\mathsf{CC}^0[p] \circ \text{MOD}_m$ circuits) require $2^{\Omega(n)}$ size to compute the AND function, where p is a prime and m is an arbitrary composite, and that only certain restricted symmetric functions could be computed in subexponential-size and depth-2 [55]. Theorem 7.1.1 shows that one additional layer of MOD_p gates makes such circuits much more powerful.

It is well-known that for distinct primes p, q, every symmetric function on n bits has a MOD_{pq} circuit of size $\exp(O(n^\varepsilon))$ and depth $O(1/\varepsilon)$.[3] Our result shows the depth can always be made 3, at the cost of increasing the modulus to a large enough constant. Hansen [57], building on Bhatnagar, Gopalan, and Lipton [22], shows that for m which is the product of r primes, and sufficiently small ℓ (smaller than each of the prime factors of m), the MOD_ℓ function can be represented by a polynomial over $\mathbb{Z}_m$ of degree $O(n^{1/r})$. As a corollary of Hansen's work, Gopalan observed [51] that for every $\varepsilon > 0$ there is an m such that the MOD_2 function has depth-3 MOD_m circuits of size 2^{n^ε}. This naturally suggests the question of whether every symmetric function admits such a circuit, which is answered by our Theorem 7.1.1.

As we allow larger depths, we can obtain MOD_m circuits with an interesting size-depth tradeoff.

Theorem 7.1.2 *Let $d \geq 3$ be an integer, and let m be a product of $r \geq 2$ distinct primes. Then every symmetric function on n bits can be computed by depth-d MOD_m circuits of size $\exp(\tilde{O}(n^{1/(r+d-3)}))$.*

To contrast, recall that the lower bounds for AC^0 are $\exp(\Omega(n^{1/(d-1)}))$ size for depth d [61], and the lower bounds for $\mathsf{AC}^0[p^k]$ (with AND, OR, and MOD_{p^k} gates) are $\exp(\Omega(n^{1/(2d)}))$ [90, 98] for prime power p^k (where the constant factor depends on p^k). Thus for constant moduli m with enough prime factors, one can beat both lower bounds with MOD_m gates.

[3] The authors don't know the origin of this construction. It follows from the fact that every function on k bits has a depth-2 MOD_{pq} circuit of size $2^{O(k)}$, and that symmetric functions can be easily "decomposed" into smaller functions (as in [4]).

For large enough depth d, we can achieve even smaller circuits with a size bound of the form $\exp(n^{O(1)/(r \cdot d)})$, multiplying r and d in the denominator, instead of adding them.

Theorem 7.1.3 *There is a universal constant $c \geq 1$ such that, for all sufficiently large depths d, and all composite m with r prime factors, every symmetric function can be computed by a MOD_m gate circuit of depth d and size $\exp(O(n^{c/((d-c)(r-1))}))$.*

We remark that the constant c in the above construction is not terribly small. (Our c is at least 6; this matters if one cares about very small d and r.) In concurrent (very recently released) work, [64] give a circuit construction with a similar tradeoff (but better constants) for the special case of the AND function, building on the polynomials of [16].

Even Smaller Circuits in ACC0. Allowing AND and OR gates in our circuit, the size of our circuit constructions can be even further improved.

Definition 7.1.1 *Say that a product m of primes $q_1, \ldots, q_r$ is* good *if every prime factor of $\phi(m)$ divides m, where ϕ is Euler's totient function.*

We note that the primorial $m = p_r\#$, the product of the first r primes, is good.[4]

Theorem 7.1.4 *Let m be a good product of r primes. For every symmetric function f on n inputs and every depth $d \geq 4$ congruent to 1 modulo 3, there exists an $\mathsf{AC}^0[m]$ circuit of depth d and size $\exp(\tilde{O}(n^{3/((r+3)(d-1)-3)}))$ computing f.*

In the proof of Theorem 7.1.4, we make use of several tools from the recent $\mathsf{AC}^0[2]$ circuits of [85] (circuits for elementary symmetric polynomials and circuits for the coin problem), along with known results on computing elementary symmetric polynomials modulo a prime.

Applying standard tricks (seen in [59, 104, 85]), Theorem 7.1.4 extends to linear threshold functions.

[4]Indeed, for all $i = 1, \ldots, r$, the prime factors of $q_i - 1$ are contained in $\{q_1, \ldots, q_{i-1}\}$, all of which divide $m = p_r\#$.

Corollary 7.1.1 *Let m be a good product of r primes. For every linear threshold function f on n inputs and every depth $d \geq 4$ congruent to 1 modulo 3, there exists an $\mathsf{AC}^0[m]$ circuit of depth $d+2$ and size $\exp(\tilde{O}(n^{3/((r+3)(d-1)-3)}))$ computing f.*

The corollary follows directly from the fact that every linear threshold function can be written as an OR of poly(n) ANDs of poly(n) symmetric functions on n-bit inputs [59]. In general, Theorem 7.1.4 implies that TC^0 circuits (composed of MAJ and NOT gates) with small fan-in also have a nontrivial simulation.

Corollary 7.1.2 *Every TC^0 circuit of depth e in which every gate has fan-in at most s has an equivalent MOD_m circuit of depth $d \cdot e$ and size at most $\exp(\tilde{O}(s^{3/((r+3)(d-1)-3)}))$, where m is a good product of r primes.*

The corollary follows from direct substitution of each MAJ gate with depth-d circuits from our Theorem 7.1.4. Note that such depth-e TC^0 circuits (where every gate has fan-in at most s) have at most $O(s^{e-1})$ gates: a depth-1 circuit has $O(1)$ gates, a depth-2 circuit has $O(s)$ gates, and so on. (We do not count inputs as gates.)

Can't you do any better? Theorem 7.1.4 shows that for certain m which are products of r primes, one can compute arbitrary symmetric functions in depth d and size $\exp(n^{\frac{c}{rd}})$ where $c > 0$ is a constant. We give evidence that it may be difficult to improve asymptotically on the dependence of r and d in the exponent of n, based on a natural hypothesis regarding TC^0 circuits, which are constant-depth circuits composed of MAJ and NOT gates. (Of course it is difficult to prove anything unconditional here, because as far as we know, polynomial-size depth-3 MOD_6 circuits could compute every EXP function. Thus we settle for conditional hardness.)

Recall that a SYM$\circ$AND circuit is a depth-2 circuit where the output is a symmetric function and the bottom layer computes ANDs of input variables and negations. The hypothesis is that subexponential-size SYM$\circ$AND circuits cannot compute TC^0 circuits in which each gate has linear fan-in.

Hypothesis 7.1.1 (SYM ∘ AND Hypothesis) *There are constants $c, k > 1$ such that for sufficiently large n, there is a function $f : \{0,1\}^n \to \{0,1\}$ computable by TC^0 circuits of depth c with at most $\tilde{O}(n)$ gates where each gate has fan-in $\tilde{O}(n)$, such that f does not have an $\exp(O(n^{1/k}))$ size* SYM ∘ AND *circuit.*

A well-known result in circuit complexity is that every ACC^0 circuit of size s can be simulated by a SYM ∘ AND circuit of size $s^{\mathrm{poly}(\log s)}$ [21, 39]. Therefore, the SYM ∘ AND Hypothesis is a strengthening of the longstanding hypothesis that $\mathsf{TC}^0 \not\subset \mathsf{ACC}^0$: the SYM ∘ AND Hypothesis implies exponential lower bounds for simulating TC^0 circuits with ACC^0 circuits. Indeed, the hypothesis implies that our ACC^0 circuits for symmetric functions are nearly size-optimal in their dependence on depth and modulus.

Theorem 7.1.5 (Near-Optimality Modulo a Conjecture) *Assuming the* SYM∘ AND *Hypothesis, there is a fixed $\alpha > 0$ such that for* every *m and d, every depth-d $\mathsf{ACC}^0[m]$ circuit computing the* MAJ *function on n inputs requires size at least $\exp(n^{\frac{\alpha}{rd}})$ for sufficiently large n, where r is the number of distinct prime factors of m.*

The proof of Theorem 7.1.5 is in Section 7.5. Therefore, we view size bounds of the form $\exp(n^{1/\Theta(rd)})$ (as seen in our results) as a natural barrier to better upper bounds on MOD_m circuits: any function with significantly smaller MOD_m circuit complexity (as a function of n, r, and d) would also yield a highly non-trivial SYM ∘ AND circuit simulation of TC^0. In order to achieve significantly smaller circuits as a function of n, d, and r, one has to at least refute the hypothesis. Of course, even assuming Hypothesis 7.1.1, our circuits can probably be improved by constant factors in the exponents.

7.1.1 Intuition

Our circuit constructions use several tricks from the literature in new ways. Here, we give a short high-level exposition of subexponential-size depth-3 circuits for computing

symmetric functions that use only MOD_m gates for composite m (Theorem 7.1.1). For simplicity, we will focus on computing EMAJ ("exact majority") functions, which output 1 on a vector $(x_1, \ldots, x_n)$ if and only if

$$\sum_i x_i = T$$

for some target $T \in \{0, 1\}$.

Our first idea is to split the set of input variables into many parts; this is taken from the folklore depth-3 AC^0 circuits for symmetric functions of size $2^{\tilde{O}(\sqrt{n})}$ (although we will beat that size bound considerably). Letting $\delta \in (0, 1)$ be a parameter, we partition the inputs $x_1, \ldots, x_n$ into $t := \lceil n^\delta \rceil$ groups $G_1, \ldots, G_t$ of $O(n^{1-\delta})$ inputs each. Our circuit will try all possible $T_1, \ldots, T_t \in \{0, 1, \ldots, n/t\}$ such that $\sum_j T_j = T$, outputting 1 if the circuit finds T_j's such that for all $j = 1, \ldots, t$, the sum of all variables in group G_j equals T_j. In a depth-3 AC^0 circuit, we can set $\delta = 1/2$ and obtain a circuit of $2^{\tilde{O}(\sqrt{n})}$ size: in particular, we take an OR over all $2^{\tilde{O}(\sqrt{n})}$ choices $(T_1, \ldots, T_t)$, take an AND over all groups $j = 1, \ldots, t$, then determine whether the sum of variables in G_j equals T_j using a CNF of size $2^{O(\sqrt{n})}$. Thus we have a circuit of type

$$\text{OR} \circ \text{AND} \circ \text{OR}$$

which computes the EMAJ function in size $2^{\tilde{O}(\sqrt{n})}$.

Using MOD_m gates where m has many distinct prime factors, we can do much better. Applying Lucas' Theorem (Theorem 7.2.1) in a new way, we construct a polynomial P_{T_j} of degree $O(n^{(1-\delta)/r})$ that can determine whether the sum of variables in group G_j equals T_j, where r is the number of distinct prime factors of m. Theorem 7.3.2 demonstrates this claim in the case of $r = 2$. (We also need another technical condition on m for this to work, but we ignore that issue here.) This polynomial can be directly simulated by a depth-2 circuit of MOD_{pm} gates (where p is a new prime that does not divide m) and $\exp(\tilde{O}(n^{(1-\delta)/r}))$ size, which can determine whether or not the sum of variables in group G_j equals the target T_j.

Substituting this depth-2 circuit into the depth-3 AC^0 circuit described above (for each group j), we would have a circuit of the form:

$$\textsc{Or of}\exp(\tilde{O}(n^{\delta}))\ \textsc{Ands of}\ \tilde{O}(n^{\delta})\ \textsc{Mod}_{pm}\ \text{of}\ \exp(\tilde{O}(n^{(1-\delta)/r}))\ \textsc{Mod}_{pm}.$$

Then, setting δ so that $\delta = (1-\delta)/r$, we obtain $\delta = 1/(r+1)$ and a circuit of $\exp(\tilde{O}(n^{1/(r+1)}))$ size. To reduce the depth down to three and use only $\textsc{Mod}_{pm}$ gates, we observe that at most one of the wires into the Or can be true, and we apply another known translation (Proposition 7.2.2) to convert the And of $\tilde{O}(n^{\delta})$ $\textsc{Mod}_{pm}$ gates into a linear sum modulo pm of $\exp(\tilde{O}(n^{\delta}))$ $\textsc{Mod}_{pm}$ gates, collapsing the $\textsc{Or} \circ \textsc{And} \circ \textsc{Mod}_{pm} \circ \textsc{Mod}_{pm}$ circuit into a depth-3 circuit of only $\textsc{Mod}_{pm}$ gates and $\exp(\tilde{O}(n^{1/(r+1)}))$ size. Setting m so that r is arbitrarily large, we obtain Theorem 7.1.1.

To obtain our stronger and more general results (Theorems 7.1.2, 7.1.3, and 7.1.4), we rely on even more tools and tricks, most of which are recorded in the next section. (Throughout the chapter, we try to state clearly which ideas are due to prior work, and which are new.)

7.2 Preliminaries

Notation 7.2.1 *For a binary vector* $\mathbf{x}$*, we use* $|\mathbf{x}|_1$ *to denote the* ℓ_1*-norm, i.e., the number of ones in* $\mathbf{x}$*.*

Besides AC^0, ACC^0, CC^0, and TC^0, we also use the following additional notation for various circuit types, all of which is standard:

Definition 7.2.1 (Sym) *A circuit of type* Sym *is a symmetric Boolean function.*

Definition 7.2.2 (EMaj) *An* EMaj *function outputs* 1 *on an input* $(x_1, \ldots, x_n) \in \{0,1\}$ *if and only if* $\sum_i x_i = T$ *for a fixed target* T*.*

It is not hard to see that, by substituting 0/1 constants appropriately, such a function can always be implemented by a gate of $O(n)$ inputs which outputs 1 if and only if

$\sum_i x_i = n/2$. This is why such gates are called "exact majority" (EMAJ).

We also make the standard assumption that all gates are allowed to include 0/1 constants in their inputs.

The following basic fact is useful to keep in mind; we will apply it frequently.

Proposition 7.2.1 *For all positive $m, n \in \mathbb{Z}$, any MOD_m gate of fan-in t can be simulated by a MOD_{mn} of fan-in nt.*

Proof. For any positive integer t, $m \mid t$ if and only if $nm \mid nt$. So $\text{MOD}_m(x_1, \ldots, x_t) = \text{MOD}_{mn}(n \cdot x_1, \ldots, n \cdot x_t)$. □

We make use of several known results. First, we note that AND circuits of small fan-in have efficient depth-2 MOD_m circuits. A version was first used in [17] in the context of MOD circuits, and more recently a strengthening was used to reduce the size-depth tradeoff for simulating ACC^0 circuits with $\text{SYM} \circ \text{AND}$ circuits [39]. (Chen and Papakonstantinou [39] call this "linearization".)

Proposition 7.2.2 ([17, 39]) *Let $a, b \geq 2$ be fixed integers with $gcd(a, b) = 1$. Every AND of k MOD_b gates can be represented by an $\text{MOD}_a \circ \text{MOD}_b$ circuit of $O(b^k)$ gates. Furthermore, on all k-bit inputs, the sum of the inputs to the output gate of the circuit is always $0 \pmod a$ or $1 \pmod a$.*

Our next tool is an old number-theoretic theorem on elementary symmetric polynomials modulo p, masterfully applied by Beigel, Barrington, and Rudich [16] in their non-trivial degree polynomials for the OR functions over composite moduli.

Theorem 7.2.1 (Lucas' Theorem [75]) *For all primes p and natural numbers n,*

$$\binom{n}{p^i} \bmod p$$

is the i-th digit in the p-ary representation of n.

Lucas' theorem has the following direct consequence for polynomial representations of Boolean functions.

Lemma 7.2.1 ([16]) *Let p be a prime, let n be a natural number, and let $e_i(\mathbf{x})$ denote the i-th elementary symmetric polynomial on n variables. For a binary vector $\mathbf{x}$, let*

$$\sum y_i \cdot p^i = |\mathbf{x}|_1$$

be the p-ary expansion of $|\mathbf{x}|_1$. Then for every i, $e_{p^i}(\mathbf{x}) \equiv y_i \mathrm{mod} p$.

In order to apply the elementary symmetric polynomials, our construction also involves arithmetic circuits over prime fields. These circuits will be translated into Boolean circuits with MOD_m gates. Such circuits were also used by [85] in their improved $\mathsf{AC}^0[2]$ circuits for MAJ.

Lemma 7.2.2 ([40, 85]) *Let p be a prime, let $n, i \in \mathbb{N}$, and let $d \geq 2$ be even. There is an arithmetic circuit over $\mathbb{F}_p$ of depth d and size $n^{O(i^{2/d})}$ computing the i-th elementary symmetric polynomial (over $\mathbb{F}_p$) on n inputs, where the output gate is a $\times$ gate.*

We also use AC^0 circuits for the coin problem. These were also used by [85] in their improved $\mathsf{AC}^0[2]$ circuits for symmetric functions.

In the following, we let $i, j \in \{0, 1, \ldots, n\}$, and let $D_{i,j}$ be any partial function satisfying the properties:

$$D_{i,j}(\mathbf{x}) = 1 \text{ if } |\mathbf{x}|_1 = i, \text{ and}$$

$$D_{i,j}(\mathbf{x}) = 0 \text{ if } |\mathbf{x}|_1 = j.$$

Lemma 7.2.3 ([84, 8, 85]) *Let $d \geq 2$ and n be natural numbers, and let $i \neq j$. Then there is an AC^0 circuit of depth d and size $\exp(O(d\,(n/|i-j|)^{1/(d-1)}))$ computing $D_{i,j}$ on n inputs, where the output gate is an* AND.

Intuitively, Lemma 7.2.3 will be useful when $|i - j|$ is "large".

7.3 CC0 Circuits for Symmetric Functions

We begin by giving efficient depth-3 CC^0 circuits for symmetric functions.

Reminder of Theorem 7.1.1 *For every $\varepsilon > 0$, there is a modulus $m \leq (1/\varepsilon)^{2/\varepsilon}$ such that every symmetric function on n bits can be computed by depth-3 MOD_m circuits of $\exp(O(n^{\varepsilon}))$ size. In fact, the circuits have the form $\text{MOD}_{p_1} \circ \text{MOD}_{p_2 \cdots p_r} \circ \text{MOD}_{p_1}$, where $p_1, \ldots, p_r$ are distinct primes.*

After that, we will generalize the result to a size-depth tradeoff in the next subsection. That tradeoff will be further improved in Section 7.4 when we allow the use of AND and OR gates.

As a warm-up, we first consider the special case where $\varepsilon > 1/3$ and $m = 30$.

Theorem 7.3.1 *Every symmetric Boolean function on n variables has a depth-3 circuit of the form $\text{MOD}_5 \circ \text{MOD}_6 \circ \text{MOD}_5$, of size $\exp(O(n^{1/3} \log n))$. Furthermore, the output gate is a linear sum which always evaluates to either 0 or 1 modulo 5.*

Note that the upper bound of Theorem 7.3.1 already beats the well-known lower bounds for depth-3 AC^0 [61]. The remainder of this section is devoted to the proof. A key component is a low-degree multivariate polynomial over $\mathbb{Z}_6$ that vanishes on a Boolean vector if and only if the sum of the ones in the vector equals a particular value.

Theorem 7.3.2 *For every $n \in \mathbb{N}$ and every $T \in \{0, 1, \ldots, n\}$, there is a polynomial $P_T(x_1, \ldots, x_n)$ of degree at most $3\sqrt{n}$ such that for all $a \in \{0,1\}^n$, $P_T(a) = 0 \bmod 6$ if and only if $\sum_i a_i = T$.*

Proof. We want a polynomial p on n variables such that for all $y_1, \ldots, y_n \in \{0,1\}$ and $T \in \{0, 1, \ldots, n\}$,

$$p(y_1, \ldots, y_n) \equiv 0 \bmod 6 \iff \sum_i y_i = T.$$

For the elementary symmetric polynomial $e_J(y_1, \ldots, y_n)$ of degree J, and for all $a_1, \ldots, a_n \in \{0, 1\}$,

$$e_J(a_1, \ldots, a_n) = \binom{(\sum_i a_i)}{J}.$$

Thus by Lucas' Theorem (Theorem 7.2.1), $e_{p^i}(a_1, \ldots, a_n) \bmod p$ equals the i-th digit in the p-ary representation of $\sum_i a_i$.

Let s and t be integers so that $2\sqrt{n} \geq 2^s > \sqrt{n}$ and $3\sqrt{n} \geq 3^t > \sqrt{n}$.

Suppose when we write $T \in \{0, 1, \ldots, n\}$ in binary notation, the s low order bits are $b_{s-1}, \ldots, b_0$. Furthermore, when we write T in ternary notation, the t low order trits are $c_{t-1}, \ldots, c_0$.

Define the polynomials

$$p_2(y_1, \ldots, y_n) := 1 - \prod_{j=0}^{s-1} (1 - (b_j - e_{2^j}(y))) \bmod 2$$

and

$$p_3(y_1, \ldots, y_n) := 1 - \prod_{j=0}^{t-1} (1 - (c_j - e_{3^j}(y))^2) \bmod 3.$$

Note the degrees of p_2 and p_3 are $O(\sqrt{n})$. In particular, $\deg(p_2) = \sum_{j=0}^{s-1} 2^j = 2^s - 1$ and $\deg(p_3) = \sum_{j=0}^{t-1} (2 \cdot 3^j) = 2(3^t - 1)/2 = 3^t - 1$.

We observe a few properties of the polynomials p_2 and p_3:

Proposition 7.3.1 *For all $a \in \{0, 1\}^n$, $p_2(a) \equiv 0 \bmod 2$ if and only if the binary representation of $\sum_i a_i$ equals $b_{s-1} \cdots b_0$ in the last s bits. Analogously, $p_3(a) \in \{0, 1\} \bmod 3$, and $p_3(a) \equiv 0 \bmod 3$ if and only if the ternary representation of $\sum_i a_i$ equals $c_{t-1} \cdots c_0$ in the last t trits.*

Proof. We prove the proposition for p_3; the case of p_2 is analogous. Let $a \in \{0, 1\}^n$. Each difference $(c_j - e_{3^j}(a))^2$ is either 0 or 1 modulo 3, and it is 0 if and only if $c_j = e_{3^j}(a)$. Thus the product $\prod_{j=0}^{t-1} (1 - (c_j - e_{3^j}(a))^2)$ equals 1 if and only if $c_j = e_{3^j}(a)$

for all $j = 0, \ldots, t-1$, hence $p_3(a) \equiv 0 \bmod 3$ if and only if $c_j \equiv e_{3^j}(a) \bmod 3$ for all $j = 0, \ldots, t-1$. Recalling that $(e_{3^j}(a) \bmod 3)$ equals the j-th trit of $\sum_i a_i$, the result follows. $\square$

We note that in general, working modulo a prime q, we may construct a polynomial with degree $(q^t - 1)$ of the form

$$p_q(y_1, \ldots, y_n) = 1 - \prod_{j=0}^{t-1} (1 - (c_j - e_{q^j}(y))^{q-1}). \tag{7.1}$$

By the above proposition, it follows that for all $a \in \{0,1\}^n$,

$$p_2(a) \equiv 0 \bmod 2 \iff \sum_i a_i \equiv T \bmod 2^s$$

and

$$p_3(y) \equiv 0 \bmod 3 \iff \sum_i a_i \equiv T \bmod 3^t.$$

Since $\sum_i a_i$ and T are both in $\{0, \ldots, n\}$ and $2^s \cdot 3^t > n$, by the Chinese Remainder Theorem we have

$$\begin{aligned} \sum_i a_i = T &\iff (\sum_i a_i \equiv T \bmod 2^s) \wedge (\sum_i a_i \equiv T \bmod 3^t) \\ &\iff (p_2(y) \equiv 0 \bmod 2) \wedge (p_3(y) \equiv 0 \bmod 3) \\ &\iff 3p_2(y) + 2p_3(y) \equiv 0 \bmod 6. \end{aligned}$$

Thus $3p_2(y) + 2p_3(y)$ is a polynomial of degree $O(\sqrt{n})$ which equals 0 mod 6 if and only if $\sum_i y_i = T$. This completes the proof of Theorem 7.3.2. $\square$

We now proceed with the proof of Theorem 7.3.1.

Proof. Let f be a symmetric function and let $g : \{0, 1, \ldots, n\} \to \{0, 1\}$ be its companion function. That is, for every $\mathbf{x}$, $f(\mathbf{x}) = g(|\mathbf{x}|_1)$.

The output gate will be a MOD_5 gate that

- (a) sums over possible choices of $T \in \{0, 1, \ldots, n\}$ such that $g(T) = 1$ and

- (b) sums over all ways to partition T into a sum of $t = \lceil n^{1/3} \rceil$ parts $T_1, \ldots, T_t \in \{0, 1, \ldots, T\}$.

There are $2^{O(n^{1/3} \log n)}$ choices over (a) and (b). We associate each part T_i with a disjoint set S_i of at most $\lceil n^{2/3} \rceil$ variables from the input. For each of the choices from (a) and (b), we wish to verify that, for all $i = 1, \ldots, t$, the sum of all variables in S_i equals T_i. Note that there is *at most* one choice from (a) and from (b) that could possibly be consistent with the given input, so we can use a *modulo*-5 *sum*, which we denote by ΣMOD_5 (not just a MOD_5 gate) to sum over these choices. This modulo-5 sum will always be either 0 or 1 modulo 5.

By our construction of EMAJ polynomials, each sum over the set S_i of $n^{2/3}$ variables can be checked with a MOD_6 gate of $2^{O(n^{1/3})}$ fan-in, where each input to the MOD_6 gate is the output of an AND of fan-in $O(n^{1/3})$. Putting these $\text{MOD}_6 \circ \text{AND}$ circuits below each wire of the ΣMOD_5, at this point, we have a ΣMOD_5 of $2^{O(n^{1/3} \log n)}$ ANDs of fan-in $O(n^{1/3})$ of MOD_6 of fan-in $2^{O(n^{1/3})}$ of ANDs of fan-in $O(n^{1/3})$.

To eliminate the AND gates, we apply Proposition 7.2.2, yielding that an AND of f MOD_q gates can be represented by a ΣMOD_p of $O(q^f)$ MOD_q gates, as long as $\gcd(p, q) = 1$. In particular, for the "middle" ANDs we set $p = 5$ and $q = 6$, and for the "bottom" ANDs we set $p = 6$ and $q = 5$. We obtain a ΣMOD_5 of $2^{O(n^{1/3} \log n)}$ MOD_6 of fan-in $2^{O(n^{1/3})}$ of MOD_5 of fan-in $O(n^{1/3})$. □

The above construction has several interesting corollaries; here is one.

Corollary 7.3.1 *Every circuit of the form* $\text{MOD}_5 \circ \text{SYM}$ *of size* $2^{O(n^{1/3} \log n)}$ *can be simulated by a depth-*3 $\text{MOD}_5 \circ \text{MOD}_6 \circ \text{MOD}_5$ *circuit of size* $2^{O(n^{1/3} \log n)}$.

Proof. We simply replace each SYM gate (which takes n inputs) in the $\text{MOD}_5 \circ \text{SYM}$ circuit with a modulo-5 sum of $\text{MOD}_6 \circ \text{MOD}_5$ as in the previous theorem. □

We are now ready to generalize to Theorem 7.1.1.

Reminder of Theorem 7.1.1 *For every* $\varepsilon > 0$*, there is a modulus* $m \leq (1/\varepsilon)^{2/\varepsilon}$ *such that every symmetric function on* n *bits can be computed by depth-*3 MOD_m

circuits of $\exp(O(n^{\varepsilon}))$ *size. In fact, the circuits have the form* $\text{MOD}_{p_1} \circ \text{MOD}_{p_2 \cdots p_r} \circ \text{MOD}_{p_1}$*, where* $p_1, \ldots, p_r$ *are distinct primes.*

Proof. Let $\varepsilon > 0$, and let f be a symmetric function. Take $k := \lfloor 1 + 1/\varepsilon \rfloor$, let m be the product of the first k primes, and let $m' = m/2$.

We use a similar construction as in Theorem 7.3.1 to get a $\text{MOD}_2 \circ \text{MOD}_{m'} \circ \text{MOD}_2$ circuit for f.

The differences are that we partition the target $T \in \{0, 1, \ldots, n\}$ into a sum of $\lfloor n^{1/k} \rfloor$ parts where each part is over $v := n^{1-1/k}$ variables, and by using $k-1$ primes instead of two, we can obtain a polynomial for EMAJ on v variables of degree $O(v^{1/(k-1)}) \leq O(n^{1/k})$ in an analogous way.

More precisely, let $T' \in \{0, 1, \ldots, v\}$ be a target value. For the first $k-1$ odd primes $q_1, \ldots, q_{k-1}$, we take $k-1$ polynomials $p_{q_1}(x), \ldots, p_{q_{k-1}}(x)$ as defined in (7.1) such that each $p_{q_i}(x)$ has degree $(q_i)^{t_i} - 1$, where the t_i are chosen such that for all $i \in [k]$,

- $(q_i)^{t_i} = \Theta(v^{1/(k-1)})$,
- $T' \leq v < \prod_i (q_i)^{t_i}$, and
- for all $a \in \{0,1\}^v$ we have

$$p_{q_i}(a) = 0 \bmod q_i \iff \sum_i a_i \equiv T' \bmod (q_i)^{t_i}.$$

By the Chinese Remainder Theorem, and similar reasoning as in Theorem 7.3.1, there are fixed coefficients $M_i \in [m']$ such that

$$\sum_{i=1}^{k-1} M_i \cdot p_{q_i}(a) \equiv 0 \bmod m' \iff \bigwedge_{i=1}^{k-1} [p_{q_i}(a) \equiv 0 \bmod q_i] \iff \sum_i a_i = T'.$$

Thus we have a polynomial of degree $O(v^{1/(k-1)})$ that vanishes modulo m' precisely when the sum of v variables equals the target T'. Naturally we might write this polynomial as a $\text{MOD}_{m'} \circ \text{AND}$ circuit of $\exp(\tilde{O}(v^{1/(k-1)}))$ size; by replacing each AND

with a modulo-m' sum of MOD_2 gates (Proposition 7.2.2), we can express it as a $\text{MOD}_{m'} \circ \text{MOD}_2$ circuit. Our final circuit has the form $\text{MOD}_2 \circ \text{MOD}_{m'} \circ \text{MOD}_2$ and size $\exp(\tilde{O}(n^{1/k})) \leq \exp(O(n^{\varepsilon}))$. □

7.3.1 Size-Depth Tradeoff with CC0

Allowing depth d circuits for $d > 3$, the size of the above construction can be improved as a function of the number of distinct primes r in the modulus. Here we only briefly describe the construction, as the size bound will be improved significantly (as a function of d and r) in the following section.

Reminder of Theorem 7.1.2 *Let $d \geq 3$ be an integer, and let m be a product of $r \geq 2$ distinct primes. Then every symmetric function on n bits can be computed by depth-d MOD_m circuits of size $\exp(\tilde{O}(n^{1/(r+d-3)}))$.*

Proof. Let p and q be the smallest prime factors of m. We prove by induction on d that there are circuits of size $\exp(O(n^{1/(r+d-3)} \log n))$, and we prove additionally that the output gate is a MOD_p gate when d is odd and a MOD_q gate when d is even.

For $d = 3$, we use the construction of Theorem 7.1.1 to obtain a $\text{MOD}_p \circ \text{MOD}_{m/p} \circ \text{MOD}_p$ circuit of depth 3 and size $\exp(O(n^{1/r} \log n))$.

For the inductive step, we proceed similarly to the proof of Theorem 7.1.1, except that we use a MOD_p or MOD_q gate as the output gate (depending on the parity of d). We partition the target T into a sum of $t = \lceil n^{1/(r+d-3)} \rceil$ parts, where each part contains at most $\lceil n^{(r+d-4)/(r+d-3)} \rceil$ variables, and our circuit sums over all $\exp(O(t \log n))$ choices for the number of true variables in each part. Since EMAJ is a symmetric function, we can inductively compute each EMAJ on $\lceil n^{(r+d-4)/(r+d-3)} \rceil$ variables with a circuit of depth $d - 1$, as guaranteed by the inductive hypothesis. These circuits have size

$$\exp(O((n^{(r+d-4)/(r+d-3)})^{1/(r+d-4)} \log n)) \leq \exp(O(n^{1/(r+d-3)} \log n)),$$

and their output gates have fan-in $\exp(O(n^{1/(r+d-3)} \log n))$. WLOG assume d is odd.

Then the depth-$(d-1)$ circuits for EMAJ described above have the form

$$\text{MOD}_q \circ \cdots \circ \text{MOD}_p \circ \text{MOD}_{m/p} \circ \text{MOD}_p,$$

and our entire circuit has the form

$$\text{MOD}_p \circ \text{AND} \circ \text{MOD}_q \circ \cdots \circ \text{MOD}_p \circ \text{MOD}_{m/p} \circ \text{MOD}_p,$$

where the ANDs have fan-in t. As before, each $\text{AND} \circ \text{MOD}_q$ can be replaced by a $\Sigma\text{MOD}_p \circ \text{MOD}_q$ gates using Proposition 7.2.2, which only increases the circuit size by a factor of $2^{O(t)}$.

Our final circuit has depth d and size $\exp(O(n^{1/(r+d-3)} \log n))$. □

A Better Dependence on Depth and Modulus. We can give a CC^0 circuit construction with a better asymptotic tradeoff (in the double-exponent). We will keep the description of this construction brief and to the point, as its size will be further improved (replaced by better constants) in the next section, using OR and AND gates.

Reminder of Theorem 7.1.3 *There is a universal constant $c \geq 1$ such that, for all sufficiently large depths d, and m which is the product of the first r prime factors, every symmetric function can be computed by a MOD_m gate circuit of depth d and size* $\exp(O(n^{c/((d-c)(r-1))}))$.

Proof. First, we recall that every symmetric function f on n variables can be expressed as a MAJ of $O(n)$ MAJ gates over the n variables (see for example [20] for a reference). Thus it suffices to give a circuit for MAJ.

Allender and Koucky [4, Theorem 3.8] give a downward self-reduction for the MAJ function: they prove that there is a universal constant $a \geq 1$ such that for every $k \geq 1$, the MAJ function on n bits can be computed by a TC^0 circuit of depth at most ak where each MAJ gate has fan-in at most $O(n^{1/k})$. Applying these circuits to

the depth-2 TC^0 circuits described in the previous paragraph, we obtain an analogous circuit of depth $2ak$ for any given symmetric function f.

Replace each MAJ gate of fan-in at most $O(n^{1/k})$ with a depth-3 MOD_m circuit of size at most

$$\exp(\tilde{O}(n^{1/(k(r-1))})),$$

as provided by Theorem 7.1.2. (Note that each NOT gate can always be replaced by a single MOD_m gate, if we do not want to allow NOT gates in our CC^0 circuit.) This results in a circuit of depth $6ak$ and size

$$\exp(\tilde{O}(n^{1/(k(r-1))})) \leq \exp(\tilde{O}(n^{1/(k(r-1))})).$$

Thus for depths $d = 6ak$ where k is a positive integer, the size bound is at most $\exp(n^{6a/(d(r-1))})$. For depths d that are not divisible by $6a$, we can simply use the construction for $d' = 6ak$ where $d' < d < 6a(k+1)$, which has size at most $\exp(n^{6a/(d'(r-1))}) < \exp(n^{6a/((d-6a)(r-1))})$. □

The above construction is not useful for $d < 6$ and small r, which are of interest. In the next section, we will show that much better constants are obtainable in the ACC^0 setting.

7.4 Size-Depth Tradeoff With ACC0

We now turn to showing how adding AND and OR gates can help improve the circuits even further. We begin with a result using the concrete modulus 42.

Theorem 7.4.1 *For every symmetric function f on n inputs and every depth d with $d \equiv 2 \bmod 6$, there is an $\mathsf{AC}^0[42]$ circuit of depth d and size $\exp(\tilde{O}(n^{\frac{6}{13(d-2)}}))$ computing f.*

Observe that, for sufficiently large d, the circuit size of Theorem 7.4.1 already drops below Smolensky's $\mathsf{AC}^0[p^k]$ depth-d lower bound of $\exp(\Omega(n^{1/(2d)}))$ size [98] for computing MOD_q when $gcd(p,q) = 1$.

We build on the results of Oliveira *et al.* [85] for computing symmetric functions in $\mathsf{AC}^0[2]$. At a high level, we note that every symmetric function can be written as an OR of AND of (partial) functions of the form $D_{i,j}$, where

$$D_{i,j}(\mathbf{x}) = 1 \text{ if } |\mathbf{x}|_1 = i, \text{ and}$$

$$D_{i,j}(\mathbf{x}) = 0 \text{ if } |\mathbf{x}|_1 = j,$$

recalling that $|\mathbf{x}|_1$ is the number of 1's in $\mathbf{x}$. Note that $D_{i,j}$ could have *arbitrary* behavior on any other Boolean inputs.

When $|i - j|$ is large, the function $D_{i,j}$ can be simulated by the standard Coin Problem, for which there are known AC^0 circuits (see Section 7.2). When $|i - j|$ is small, we give a new construction of $\mathsf{AC}^0[42]$ circuits for $D_{i,j}$.

We will utilize arithmetic circuits for elementary symmetric polynomials. To that end, the following lemma shows how to generically translate low-depth arithmetic circuits over $\mathbb{F}_p$ into $\mathsf{AC}^0[p(p-1)]$ circuits, in a way that only increases the circuit depth by a 3/2 multiplicative factor. (Getting some constant factor increase is not too difficult; Agrawal, Allender, and Datta [1] first showed a correspondence between ACC^0 and arithmetic circuits over finite fields. Their representation of field elements in Boolean circuits does not preserve depth as well as ours, however.)

Lemma 7.4.1 *Let p be prime, and let C be an arithmetic circuit over $\mathbb{F}_p$ of size s and depth $2d$ (with alternating layers of $+$ and $\times$ gates, with $+$ at the output) on n inputs, such that for every $\mathbf{x} \in \{0,1\}^n$, $C(\mathbf{x}) \in \{0,1\}$. Then C is equivalent to an $\mathsf{AC}^0[p(p-1)]$ circuit C' of size $O(s \cdot p)$ and depth $3d$.*

Proof. We represent an element x of $\mathbb{F}_p$ in unary, by p indicator bits

$$b_0(x), b_1(x), \ldots, b_{p-1}(x),$$

where $b_0(x) = 0$ iff $x = 0$, and for $i \neq 0$, we let $b_i(x) = 1$ iff $x = i$. (We treat the 0-th indicator bit as a special case to make later constructions easier.) We now obtain C'

by replacing each gate in C with a small $\mathsf{AC}^0[p(p-1)]$ gadget circuit.

For each addition gate of C computing

$$x = \sum_{j=1}^{k} x_j,$$

we replace that gate with p parallel $\textsc{Mod}_p$ gates, so that

$$b_i(x) = 1 \iff (p-i) + \sum_{j=1}^{k} \sum_{i'=1}^{p-1} i' \cdot b_{i'}(x_j) \equiv 0 \bmod p.$$

(As a special case, we output the negation of the right hand side in the case of $b_0(x)$.) To see why this works, we observe that the inner sum computes x_j, and so the outer sum computes x. Now $x + (p-i) \equiv 0 \bmod p$ precisely when $x = i$.

Take g to be a generator of the multiplicative group $\mathbb{F}_p^*$ of $\mathbb{F}_p$, and let $\log_g(n)$ denote the discrete logarithm base g in $\mathbb{F}_p$ (i.e., $g^{\log_g(n)} = n \bmod p$). For each multiplication gate of C computing

$$x = \prod_{j=1}^{k} x_j,$$

we replace that gate with an $\textsc{And}$ gate placed in parallel with $p-1$ $\textsc{And} \circ \textsc{Mod}_{p-1}$ circuits, implementing the conditions

$$b_0(x) = \bigwedge_{j=1}^{k} b_0(x_j),$$

and for $i \neq 0$,

$$b_i(x) = G_i \wedge \bigwedge_{j=1}^{k} b_0(x_j).$$

where G_i is a $\textsc{Mod}_{p-1}$ gate such that

$$G_i = 1 \iff (p - \log_g(i)) + \sum_{j=1}^{k} \sum_{i'=2}^{p-1} b_{i'}(x_j) \cdot \log_g(i') \equiv 0 \bmod p-1.$$

To see why this works, we observe that the inner sum computes the discrete logarithm

of x_j (for the same reason that the inner sum in the addition case computes x_j). Since $x = \prod x_j$, we have (for non-zero x) $\log_g x = \sum \log_g x_j$, so the outer sum computes the discrete logarithm of x. Now $(\log_g x) + (p - \log_g i) \equiv 0 \bmod p-1$ precisely when $x = i$.

Finally, we take the output wire of C' to be the negation of the b_0 wire from the output gate of C. □

We note that as a special case, an arithmetic circuit over $\mathbb{F}_2$ can be viewed directly as an $\mathsf{AC}^0[2]$ circuit (with the same size and depth), since an element of $\mathbb{F}_2$ is simply a bit, addition in $\mathbb{F}_2$ is MOD_2, and multiplication is AND. Additionally, when $p-1$ is not square-free, we can improve the modulus in the circuit above.

Lemma 7.4.2 *Let p be prime, and let C be an arithmetic circuit over $\mathbb{F}_p$ of size s and depth $2d$ (with alternating layers of $+$ and $\times$ gates, with $+$ at the output) on n inputs, such that for every $\mathbf{x} \in \{0,1\}^n$, $C(\mathbf{x}) \in \{0,1\}$. Then C is equivalent to an $\mathsf{AC}^0[pm]$ circuit of size $O((sp)^{p-1})$ and depth $3d$, where m is the product of the distinct prime factors of $p-1$.*

Proof. We start with the circuit C' given by Lemma 7.4.1. We now use Theorem 7.2.1 (Lucas) and the Chinese Remainder Theorem to simulate each MOD_{p-1} gate of fan-in f using an $\text{AND} \circ \text{MOD}_m \circ \text{AND}$, as follows. We observe that

$$\sum_{i=1}^{f} y_i \equiv 0 \bmod p-1 \iff \bigwedge_q \bigwedge_k \left[\sum_S \prod_{i \in S} y_i \equiv 0 \bmod q \right], \tag{7.2}$$

where the outermost AND ranges over all primes q dividing $p-1$, the inner AND ranges over all $k \geq 0$ such that q^k (strictly) divides $p-1$, and the summation ranges over all subsets $S \subseteq [f]$ of size q^k.

In particular, letting $p - 1 = q_1^{k_1} \cdots q_t^{k_t}$, $\sum_i y_i$ is divisible by $p-1$ if and only if it is divisible by $q_i^{k_i}$ for all i, by the Chinese Remainder Theorem. Theorem 7.2.1 says we can check that $\sum_i y_i$ is divisible by $q_i^{k_i}$, by checking that $\binom{\sum_i y_i}{q_i^j}$ is divisible by q, for all $j = 0, \ldots, k_i - 1$. Equation (7.2) is checking precisely these conditions.

For each q, k, the condition in square brackets is a homogeneous polynomial of degree at most $p-1$, and the degree of the polynomial is different for each choice of q and k. Therefore the total number of monomials over all such polynomials is at most

$$\sum_{i=1}^{p-1} \binom{f}{i} \leq f^{p-1}.$$

Each monomial can be thought of an AND gate in the natural way. The two outer ANDs (over q and k) can be collapsed into a single AND of fan-in at most $\log(p-1)$. Recall that C' from Lemma 7.4.1 is of the form

$$(\text{AND} \circ \text{MOD}_{p-1} \circ \text{MOD}_p)^d.$$

Therefore the bottom layer of AND gates (the monomials above) have MOD_p gates as inputs. Applying 7.2.2, each of these ANDs can be replaced by a sum (mod m) of fan-in $O(p^{p-1})$ which is Boolean-valued. These sums can be absorbed into the layer of MOD_m gates. Since the MOD_{p-1} gates in C' can only be inputs to AND gates, the top layer of AND gates in our circuit for (7.2) can be absorbed into the AND gates for which they are inputs. Our final circuit has the form

$$(\text{AND} \circ \text{MOD}_m \circ \text{MOD}_p)^d.$$

Note the above construction increases the size to at most $O((sp)^{p-1})$. □

Putting these results together, we obtain the following:

Theorem 7.4.2 *Let d be a multiple of 6, let n be a natural number, and let $\alpha \in (0, 1]$. Set*

$$s := \left\lceil \frac{3\alpha \log_2 n}{7} \right\rceil, \; t := \left\lceil \frac{2\alpha \log_3 n}{7} \right\rceil, \; u := \left\lceil \frac{2\alpha \log_7 n}{7} \right\rceil,$$

and $m := 2^s 3^t 7^u$. Then there is an $\mathsf{AC}^0[42]$ circuit of depth $d+1$ and size $2^{\tilde{O}(n^{6\alpha/7d})}$ computing the MOD_m function on n inputs, where the output gate is an AND gate.

Proof. Applying Lemma 7.2.2, we construct:

- arithmetic circuits $C_1, C_2, \ldots, C_{2^s}$ over $\mathbb{F}_2$ of depth d, where C_i computes the i-th elementary symmetric polynomial modulo 2,

- arithmetic circuits $D_1, D_3, \ldots, D_{3^t}$ over $\mathbb{F}_3$ of depth $2d/3$, where D_i computes the i-th elementary symmetric polynomial modulo 3, and

- arithmetic circuits $E_1, E_7, \ldots, E_{7^u}$ over $\mathbb{F}_7$ of depth $2d/3$, where E_i computes the i-th elementary symmetric polynomial modulo 7,

all of which have size $n^{O(n^{6\alpha/7d})}$, given our parameters.

We convert each of the D_i and E_i into $\mathsf{AC}^0[42]$ circuits D'_i and E'_i using Lemma 7.4.1, and as previously observed, the C_i are already $\mathsf{AC}^0[2]$ circuits.

Finally, from Lemma 7.2.1 and the Chinese Remainder Theorem, all of the $C_i(\mathbf{x})$, $D'_i(\mathbf{x})$, and $E'_i(\mathbf{x})$ output 1 if and only if $|\mathbf{x}|_1 \equiv 0 \bmod m$. Our final circuit for $\textsc{Mod}_m$ is obtained by taking the AND of all of these circuits. □

We are now ready to prove Theorem 7.4.1.

Proof. Let $d \equiv 2 \bmod 6$, let f be a symmetric function on n inputs, and let g be its companion function; that is, for every $\mathbf{x}$, $f(\mathbf{x}) = g(|\mathbf{x}|_1)$. We begin with the same opening move as Oliveira, Santhanam, and Srinivasan [85], observing that

$$f(\mathbf{x}) = \bigvee_{i \in g^{-1}(1)} \bigwedge_{j \neq i} D_{i,j},$$

where $D_{i,j}(\mathbf{x}) = 1$ if $|\mathbf{x}|_1 = i$ and $D_{i,j}(\mathbf{x}) = 0$ if $|\mathbf{x}|_1 = j$ (and has otherwise arbitrary behavior). Thus it suffices to construct circuits $C_{i,j}$ computing functions consistent with $D_{i,j}$.

When $|i - j| \geq n^{7/13}$, Lemma 7.2.3 gives an AC^0 circuit $C_{i,j}$ of depth $d-1$ and size $\exp(\tilde{O}(n^{6/(13(d-2))}))$ computing $D_{i,j}$.

When $|i - j| \leq n^{7/13}$, we observe that a circuit for $\textsc{Mod}_m$ suffices, with $m > n^{7/13}$. We take $\alpha = 7/13$ in Theorem 7.4.2. Then we have a circuit $C'_{i,j}$ of depth $d-1$ and size $\exp(\tilde{O}(n^{6/(13(d-2))}))$ computing the $\textsc{Mod}_m$ function on $2n$ inputs, where $m > n^{7/13}$.

We now take $C_{i,j}(\mathbf{x}) = C'_{i,j}(\mathbf{x}1^{m-i}0^{n-m+i})$. Finally, we set

$$C = \bigvee_{i \in g^{-1}(1)} \bigwedge_{j \neq i} C_{i,j}.$$

We can collapse the output AND gates of all of the $C_{i,j}$ into the second layer AND gates, so C has depth d and size $\exp(\tilde{O}(n^{6/(13(d-2))}))$, as desired. □

More generally, for certain m which are the product of r primes, we can improve the results of Theorem 7.4.1. Recall from the introducion that we defined a product m of primes $q_1, \ldots, q_r$ to be **good** if every prime factor of $\phi(m)$ divides m, and we noted that the primorial $m = p_r\#$, the product of the first r primes, is good.

Reminder of Theorem 7.1.4 *Let m be a good product of r primes. For every symmetric function f on n inputs and every depth $d \geq 4$ congruent to 1 modulo 3, there exists an $\mathsf{AC}^0[m]$ circuit of depth d and size $\exp(\tilde{O}(n^{3/((r+3)(d-1)-3)}))$ computing f.*

Proof. Let

$$m = \prod_{a=1}^{r} p_a$$

be a good product of r primes. For each $a \in [r]$, let $s_a = \lceil \alpha \log_{p_a} n \rceil$ for some α to be defined later. By Lemma 7.2.2, there are arithmetic circuits $C_{a,b}$ over $\mathbb{F}_{p_a}$ of depth $(2/3)(d-1)$ and size $n^{O(n^{3\alpha/(d-1)})}$ computing the p_a^b-th elementary symmetric polynomial in $2n$ inputs over $\mathbb{F}_{p_a}$. By Lemma 7.4.2, we can convert these into $\mathsf{AC}^0[m]$ circuits $C'_{a,b}$ of depth $d-1$ and size $2^{\tilde{O}(n^{3\alpha/(d-1)})}$. When $|i-j| \leq n^{\alpha r}$, $i \not\equiv j \bmod p_a^{s_a}$ for at least one a by the Chinese Remainder Theorem, so we can construct a circuit $E_{i,j}$ computing $D_{i,j}$ by taking

$$E_{i,j}(\mathbf{x}) = \neg C'_{a,b}(\mathbf{x}, 0^{p_a^{s_a}-j}1^{n+j-p_a^{s_a}})$$

for some pair (a,b). When $|i-j| \geq n^{\alpha r}$, we use Lemma 7.2.3 to get a circuit $E_{i,j}$ of depth $d-1$ and size $\exp\left(\tilde{O}\left(n^{(1-\alpha r)/(d-2)}\right)\right)$ computing $D_{i,j}$. All of the $E_{i,j}$ have AND gates as output gates, so we take $\alpha = \dfrac{d-1}{(r+3)(d-1)-3}$ to balance the sizes of the two circuit constructions and complete the proof as per Theorem 7.4.1. □

It is worth noting that when $2 \mid m$, we can improve this construction slightly. When $2 \mid m$ (and $6 \mid d-1$), the $(r+3)(d-1)-3$ in the denominator of the double exponent instead becomes $\left(r + \frac{7}{2}\right)(d-1) - 3$.

7.5 Sym ∘ And Hypothesis

Let us recall the Sym∘And Hypothesis and its consequence stated in the introduction.

Reminder of Hypothesis 7.1.1 *There are constants $c, k > 1$ such that for sufficiently large n, there is a function $f : \{0,1\}^n \to \{0,1\}$ computable by TC^0 circuits of depth c with at most $\tilde{O}(n)$ gates where each gate has fan-in $\tilde{O}(n)$, such that f does not have an $\exp(O(n^{1/k}))$ size* Sym ∘ And *circuit.*

Reminder of Theorem 7.1.5 *Assuming the* Sym∘And *Hypothesis (Hypothesis 7.1.1), there is a fixed $\alpha > 0$ such that for* every *m and d, every depth-d $\mathsf{ACC}^0[m]$ circuit computing the* Maj *function on n inputs requires size at least $\exp\left(n^{\frac{\alpha}{rd}}\right)$ for sufficiently large n, where r is the number of distinct prime factors of m.*

We prove the contrapositive. We start with the negation of the theorem's conclusion:

> Suppose for every $\alpha > 0$, there is some modulus m which is a product of r primes, along with some depth d, such that Maj can be computed by a depth-d $\mathsf{ACC}^0[m]$ circuit of size $\exp\left(O\left(n^{\frac{\alpha}{rd}}\right)\right)$.

Assuming the above, we will refute the Sym ∘ And Hypothesis: we will show for all $c, k > 1$ and every function f computable by the appropriate depth-c TC^0 circuits, f has an $\exp\left(O\left(n^{1/k}\right)\right)$ size Sym ∘ And circuit.

Let $c, k > 1$ be arbitrary. Let C be a TC^0 circuit C with depth c and $\tilde{O}(n)$ gates each of fan-in at most $\tilde{O}(n)$. Suppose we substitute each Maj gate of C with a copy of the assumed $\mathsf{ACC}^0[m]$ circuit. We obtain a $\mathsf{ACC}^0[m]$ circuit C' of depth at most $c \cdot d$ and of size $\exp(\tilde{O}(n^{\frac{\alpha}{rd}}))$ such that C' is equivalent to C.

Chen and Papakonstantinou [39] prove that for every depth-d' size-s circuit D over AND, OR, and MOD_m gates, where m is the product of r distinct primes, D is equivalent to a SYM $\circ$ AND circuit D' of size at most

$$S'(s, m, r, d') = 2^{(m \log s)^{10rd'}}.$$

Applying their reduction to our C', we obtain a SYM $\circ$ AND circuit C'' of size $\exp(\tilde{O}(n^{10\alpha c}))$ that is equivalent to our original circuit C. For all $\alpha < 1/(10ck)$, we obtain a SYM $\circ$ AND circuit equivalent to C with size $\exp(O(n^{1/k}))$.

7.6 Conclusion

We believe our work demonstrates that CC^0 circuits are not as weak as conventional wisdom anticipates, even at depth three. We hope that researchers seriously consider (possibly refuting) the SYM $\circ$ AND hypothesis, as it stands in the way of obtaining significantly smaller CC^0 and ACC^0 circuits for symmetric functions.

A natural next step would be to explore how much further our constructions can be pushed beyond symmetric functions. Our Theorem 7.1.5 already demonstrates that TC^0 circuits with linearly many gates and linear fan-in can be non-trivially simulated with CC^0 circuits in subexponential size. Another question is whether NC^1 circuits or Boolean formulas can be simulated similarly. For another example, it is well-known that time t and space s computations can be simulated with depth-3 AC^0 circuits of size $2^{O(\sqrt{t \cdot s})}$; this follows from efficient simulations in the polynomial hierarchy of space-bounded computation [80]. Could the size of this construction be improved, using MOD_m gates? If such an improved circuit could be constructed in a uniform way, it would likely imply new time-space lower bounds for decision problems in PP or the counting hierarchy [5]. However, even a non-uniform construction would be very interesting.

Bibliography

[1] Manindra Agrawal, Eric Allender, and Samir Datta. On TC^0, AC^0, and arithmetic circuits. *J. Comput. Syst. Sci.*, 60(2):395–421, 2000.

[2] Miklós Ajtai. Σ^1_1-formulae on finite structures. *Annals of pure and applied logic*, 24(1):1–48, 1983.

[3] Eric Allender and Vivek Gore. A uniform circuit lower bound for the permanent. *SIAM J. Comput.*, 23(5):1026–1049, 1994.

[4] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *JACM*, 57(3), 2010.

[5] Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 295–302. IEEE Computer Society, 2001.

[6] Shaull Almagor, Brynmor Chapman, Mehran Hosseini, Joël Ouaknine, and James Worrell. Effective divergence analysis for linear recurrence sequences. In *Proceedings of the 29th International Conference on Concurrency Theory (CONCUR 2018), LIPIcs 118*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[7] Ingo Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199:339–355, 1994.

[8] Kazuyuki Amano. Bounds on the size of small depth circuits for approximating majority. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 59–70. Springer, 2009.

[9] Kazuyuki Amano. On the size of depth-two threshold circuits for the inner product mod 2 function. In Alberto Leporati, Carlos Martín-Vide, Dana Shapira, and Claudio Zandron, editors, *Language and Automata Theory and Applications*, pages 235–247, Cham, 2020. Springer International Publishing.

[10] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach*. Cambridge University Press, Cambridge, 2009.

[11] James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994.

[12] Albert Atserias. Distinguishing sat from polynomial-size circuits, through black-box queries. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 8–pp. IEEE, 2006.

[13] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991.

[14] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Annual international cryptology conference*, pages 1–18. Springer, 2001.

[15] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989. See also STOC'86.

[16] David A. Mix Barrington, Richard Beigel, and Steven Rudich. Representing Boolean functions as polynomials modulo composite numbers. *Comput. Complexity*, 4:367–382, 1994.

[17] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41, 1990.

[18] David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Inf. Comput.*, 89(2):109–132, 1990.

[19] David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC1. *J. ACM*, 35(4):941–952, 1988. See also STOC'87.

[20] Paul Beame, Erik Brisson, and Richard E. Ladner. The complexity of computing symmetric functions using threshold circuits. *Theor. Comput. Sci.*, 100(1):253–265, 1992.

[21] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, pages 350–366, 1994.

[22] Nayantara Bhatnagar, Parikshit Gopalan, and Richard J. Lipton. Symmetric polynomials over z_m and simultaneous communication protocols. *J. Comput. Syst. Sci.*, 72(2):252–285, 2006.

[23] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM (JACM)*, 65(6):39, 2018.

[24] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of the ACM (JACM)*, 42(1):269–291, 1995.

[25] Andrej Bogdanov, Kunal Talwar, and Andrew Wan. Hard instances for satisfiability and quasi-one-way functions. In *ICS*, pages 290–300, 2010.

[26] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79(4):1233–1285, 2017.

[27] Bernd Borchert and Frank Stephan. Looking for an analogue of rice's theorem in circuit complexity theory. In *Kurt Gödel Colloquium on Computational Logic and Proof Theory*, pages 114–127. Springer, 1997.

[28] Jean Bourgain. Walsh subspaces of L^p-product spaces. *Séminaire d'Analyse fonctionnelle (dit" Maurey-Schwartz")*, pages 1–14, 1979.

[29] Nader H Bshouty, Richard Cleve, Ricard Gavalda, Sampath Kannan, and Christino Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

[30] Venkatesan Chakaravarthy and Sambuddha Roy. Finding irrefutable certificates for S_2^p via arthur and merlin. In *STACS 2008*, pages 157–168. IBFI Schloss Dagstuhl, 2008.

[31] Brynmor Chapman. The gotsman-linial conjecture is false. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 692–699. SIAM, 2018.

[32] Brynmor Chapman and R. Ryan Williams. The circuit-input game, natural proofs, and testing circuits with data. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 263–270, 2015.

[33] Brynmor Chapman and R. Ryan Williams. Black-box hypotheses and lower bounds. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[34] Brynmor Chapman and R. Ryan Williams. Smaller acc0 circuits for symmetric functions. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[35] Arkadev Chattopadhyay, Navin Goyal, Pavel Pudlák, and Denis Thérien. Lower bounds for circuits with MODm gates. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 709–718. IEEE Computer Society, 2006.

[36] Arkadev Chattopadhyay and Avi Wigderson. Linear systems over composite moduli. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 43–52. IEEE Computer Society, 2009.

[37] Lijie Chen, Xin Lyu, and R. Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1–12. IEEE, 2020.

[38] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. Average-case lower bounds and satisfiability algorithms for small threshold circuits. In *Proceedings of the 31st Conference on Computational Complexity*, pages 1–35, 2016.

[39] Shiteng Chen and Periklis A. Papakonstantinou. Depth reduction for composites. *SIAM J. Comput.*, 48(2):668–686, 2019.

[40] Xi Chen, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Near-optimal small-depth lower bounds for small distance connectivity. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 612–625. ACM, 2016.

[41] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[42] Ilias Diakonikolas, Prahladh Harsha, Adam Klivans, Raghu Meka, Prasad Raghavendra, Rocco A Servedio, and Li-Yang Tan. Bounding the average sensitivity and noise sensitivity of polynomial threshold functions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 533–542, 2010.

[43] Ilias Diakonikolas, Prasad Raghavendra, Rocco A Servedio, and Li-Yang Tan. Average sensitivity and noise sensitivity of polynomial threshold functions. *SIAM Journal on Computing*, 43(1):231–253, 2014.

[44] Yuval Filmus, Hamed Hatami, Steven Heilman, Elchanan Mossel, Ryan O'Donnell, Sushant Sachdeva, Andrew Wan, and Karl Wimmer. Real analysis in computer science: A collection of open problems. *Preprint available at https://simons. berkeley. edu/sites/default/files/openprobsmerged. pdf*, 2014.

[45] Lance Fortnow, Russell Impagliazzo, Valentine Kabanets, and Christopher Umans. On the complexity of succinct zero-sum games. *Computational Complexity*, 17(3):353–376, 2008.

[46] Lance Fortnow, Aduri Pavan, and Samik Sengupta. Proving sat does not have small circuits with an application to the two queries problem. *Journal of Computer and System Sciences*, 74(3):358–363, 2008.

[47] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984. See also FOCS'81.

[48] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016.

[49] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Hiding secrets in software: A cryptographic approach to program obfuscation. *Communications of the ACM*, 59(5):113–120, 2016.

[50] Sally A Goldman and Michael J Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

[51] Parikshit Gopalan. *Computing with polynomials over composites.* PhD thesis, Georgia Institute of Technology, 2006.

[52] Parikshit Gopalan and Rocco A. Servedio. Learning and lower bounds for ac0 with threshold gates. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 588–601. Springer, 2010.

[53] Craig Gotsman and Nathan Linial. Spectral properties of threshold functions. *Combinatorica*, 14(1):35–50, 1994.

[54] Michael D Grigoriadis and Leonid G Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995.

[55] Vince Grolmusz and Gábor Tardos. Lower bounds for ($\mathrm{MOD_p}$-$\mathrm{MOD_m}$) circuits. *SIAM J. Comput.*, 29(4):1209–1222, 2000.

[56] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If np languages are hard on the worst-case, then it is easy to find their hard instances. *Computational Complexity*, 16(4):412–441, 2007.

[57] Kristoffer Arnsfelt Hansen. On modular counting with polynomials. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006)*, pages 202–212. IEEE Computer Society, 2006.

[58] Kristoffer Arnsfelt Hansen and Michal Koucký. A new characterization of ACC0 and probabilistic CC0. *Comput. Complex.*, 19(2):211–234, 2010.

[59] Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Exact threshold circuits. In *CCC*, pages 270–279, 2010.

[60] Prahladh Harsha, Adam Klivans, and Raghu Meka. Bounding the sensitivity of polynomial threshold functions. *arXiv preprint arXiv:0909.5175*, 2009.

[61] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20, 1986.

[62] Lane A. Hemaspaandra and Mayur Thakur. Lower bounds and the hardness of counting properties. *Theoretical computer science*, 326(1-3):1–28, 2004.

[63] C. Antony R. Hoare. Quicksort. *The computer journal*, 5(1):10–16, 1962.

[64] Paweł M Idziak, Piotr Kawałek, and Jacek Krzaczkowski. Complexity of modular circuits. *arXiv preprint arXiv:2106.02947*, 2021.

[65] Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, Pierre McKenzie, and Shadab Romani. Does looking inside a circuit help? In *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

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

[67] Daniel M. Kane. A structure theorem for poorly anticoncentrated gaussian chaoses and applications to the study of polynomial threshold functions. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 91–100. IEEE, 2012.

[68] Daniel M. Kane. The correct exponent for the gotsman-linial conjecture. In *2013 IEEE Conference on Computational Complexity*, pages 56–64. IEEE, 2013.

[69] Daniel M. Kane and R. Ryan Williams. Super-linear gate and super-quadratic wire lower bounds for depth-two and depth-three threshold circuits. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 633–643, 2016.

[70] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

[71] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *Journal of the ACM (JACM)*, 40(3):607–620, 1993.

[72] Richard Lipton. New directions in testing. *Distributed computing and cryptography*, 2:191–202, 1991.

[73] Richard J Lipton. *The P=NP question and Gödel's lost letter*. Springer, 2010.

[74] Richard J Lipton and Neal E Young. Simple strategies for large zero-sum games with applications to complexity theory. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 734–740, 1994.

[75] Edouard Lucas. Sur les congruences des nombres eulériens et des coefficients différentiels des fonctions trigonométriques suivant un module premier. *Bulletin de la Société Mathématique de France*, 6:49–54, 1878.

[76] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 479:480, 1969.

[77] Ketan D Mulmuley. On p vs. np and geometric complexity theory: Dedicated to sri ramakrishna. *Journal of the ACM (JACM)*, 58(2):1–26, 2011.

[78] Cody D. Murray and R. Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime from a new easy witness lemma. *SIAM J. Comput.*, 49(5), 2020.

[79] John Nash. Letter to the united states national security agency.(january 1955). *Available at NSA web site https://www.nsa.gov/news-features/declassified-documents/nash-letters/assets/files/nash_ letters1.pdf*, 1955.

[80] V. Nepomnjascii. Rudimentary predicates and Turing calculations. *Soviet Mathematics - Doklady*, 11(6):1462–1465, 1970.

[81] Ilan Newman. Private vs. common random bits in communication complexity. *Information processing letters*, 39(2):67–71, 1991.

[82] Ryan O'Donnell. Open problems in analysis of boolean functions. *arXiv preprint arXiv:1204.6447*, 2012.

[83] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[84] Ryan O'Donnell and Karl Wimmer. Approximation by DNF: examples and counterexamples. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 195–206. Springer, 2007.

[85] Igor Carboni Oliveira, Rahul Santhanam, and Srikanth Srinivasan. Parity helps to compute majority. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 23:1–23:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[86] Patrick E. O'neil. Hyperplane cuts of an n-cube. *Discrete Mathematics*, 1(2):193–195, 1971.

[87] Joël Ouaknine and James Worrell. Decision problems for linear recurrence sequences. In *Proceedings of the 6th International Workshop on Reachability Problems (RP 2012), LNCS 7550*. Springer, 2012.

[88] Joël Ouaknine and James Worrell. Ultimate positivity is decidable for simple linear recurrence sequences. In *Proceedings of 41st International Colloquium on Automata, Languages, and Programming (ICALP 2014), LNCS 8573*. Springer, 2014.

[89] Ron Patton. *Software Testing, 2nd Editon*. Sams Publishing, Indianapolis, 2005.

[90] Alexander Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[91] Alexander A Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[92] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953.

[93] Shadab Romani. *Succinct representations of Boolean functions and the Circuit-SAT problem*. PhD thesis, Memorial University of Newfoundland, 2016.

[94] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484. ACM, 2014.

[95] Grant R Schoenebeck and Salil Vadhan. The computational complexity of nash equilibria in concisely represented games. *ACM Transactions on Computation Theory (TOCT)*, 4(2):1–50, 2012.

[96] Ayumi Shinohara and Satoru Miyano. Teachability in computational learning. *New Generation Computing*, 8(4):337–347, 1991.

[97] Michael Sipser. Borel sets and circuit complexity. In *STOC*, pages 61–69, 1983.

[98] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *STOC*, pages 77–82, 1987.

[99] Howard Straubing and Denis Thérien. A note on MODp - MODm circuits. *Theory Comput. Syst.*, 39(5):699–706, 2006.

[100] Denis Thérien. Circuits constructed with MODq gates cannot compute AND in sublinear size. *Comput. Complex.*, 4:383–388, 1994.

[101] Mark Utting and Bruno Legeard. *Practical model-based testing: a tools approach*. Elsevier, 2010.

[102] Emanuele Viola. Randomness buys depth for approximate counting. *computational complexity*, 23(3):479–508, 2014.

[103] R. Ryan Williams. Nonuniform ACC circuit lower bounds. *JACM*, 61(1):2, 2014. See also CCC'11.

[104] R. Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. *Theory Comput.*, 14(1):1–25, 2018. Preliminary version in STOC'14.

[105] Andrew Chi-Chih Yao. On ACC and threshold circuits. In *FOCS*, pages 619–627, 1990.

[106] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227. IEEE Computer Society, 1977.