The Complexity of Meta-Computational Problems

Ninad Rajgopal
Magdalen College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2020

This thesis is dedicated to my parents, Soumya and Rajgopal, and my grandfather Ramanna.

Acknowledgements

First and foremost, I would like to thank my advisor Rahul Santhanam for his dedicated guidance throughout my D.Phil. I am very grateful for his enlightening advice, for sharing his deep insights in the field, and for his encouragement and support. This thesis would not be possible without his guidance, and it has been a privilege to be advised by him during this journey as a doctoral student.

I am thankful to the University of Oxford and Magdalen College for providing a supportive environment for my research. I am grateful to Rahul's ALUnif grant from the European Research Council (ERC) and the Department of Computer Science at the University of Oxford for funding me for most part of my doctoral studies. I am also very thankful to Magdalen College for generously providing financial support during the last year of my degree, and specially ensuring that I did not have to struggle during the difficult times caused by the pandemic.

I would like to thank Igor Oliveira and Ján Pich for their encouraging presence and the stimulating discussions, which helped me understand and build many of my interests in complexity theory. During my graduate studies, I had the good fortune to work with Rahul, Igor, Ján and several researchers like Srikanth Srinivasan, Lijie Chen, Shuichi Hirahara, Georgios Birmpas, Jiarui Gan, Alexandros Hollender, Francisco Marmolejo and Alexandros Voudouris. I thank them for everything I have learnt during these collaborations and for ensuring that research was a more enjoyable activity. Special thanks are due to Nikhil Balaji, Varun Kanade, Dimitrios Myrisiotis and Marco Carmosino for many helpful conversations during these last few years.

I also wish to thank my transfer and confirmation assessors Leslie Goldberg, Varun Kanade and Standa Zivný for their constructive feedback and advice. Thanks also to the staff at Magdalen and the CS department, particularly Julie Sheppard and Sarah Retz-Jones for their constant help with the admin processes.

I would like to thank N.R.Prashanth and Arnab Bhattacharyya for spurring me to pursue further study in theoretical computer science. I also want to thank Chandan Saha for introducing me to computational complexity theory through his graduate course at IISc.

Many thanks to the organisers of the St.Petersburg Complexity Semester, the Dagstuhl workshop on "Proof Complexity" and the Lower Bounds Semester at the Simons Institute, for providing me an opportunity to attend them. It has been very inspiring to listen and talk to other researchers in complexity theory.

Massive thanks to all those who have made my journey in Oxford unforgettable with their much valued friendships. To my fellow inhabitants of room 224: Francisco, Alex, Philip, Aris, Georgios, Edwin and Christian – for all the great times and your support. To Andrius, Nikola, Abhishek, and Subhayan – for making the department a great place to work in. To my housemates over the years: Rishika, Matthew, Shai, George, Adam and Josh – for making Longwall House a wholesome place to live in. To Vikranth, Urmi, Mikesh, Nikita, and Rhea – for ensuring that I was well-fed and had a home away from home. And finally, to Vikaran, Bhagya, Swaraj, Mike, Madhav, Andrea, Citlali, Poncho, Sneha, Ashwin, Mariyam, Zoe, Miriam, Kalpana, Radhika, and Akshay, for their friendship over the years.

Finally, thanks to Arushi for her patience, her support, and for her steadfast presence throughout my D.Phil. Lastly, I would like to thank my parents Soumya and Rajgopal, and my grandfather, Ramanna, for their unwavering support. I owe everything in this thesis to them.

Abstract

This thesis focuses on problems which themselves encode questions about circuits or algorithms, also called *meta-computational problems*. The thesis is centered on meta-computational problems like $\mathcal{C}$-MCSP (minimum circuit size problem), $\mathcal{C}$-Learnability and $\mathcal{C}$-Satisfiability, for some circuit class $\mathcal{C}$. We study mathematical questions pertaining to such problems, and their deep connections to the theory of lower bounds, which is a general theme that has attracted a lot of interest in recent years among complexity theorists.

We first study non-uniform lower bounds for MCSP (and its variants) motivated by its importance for the theory of *Hardness Magnification*. This phenomenon reduces major complexity separations (such as $\mathsf{NP} \not\subseteq \mathsf{NC}^1$) to proving "barely" non-trivial lower bounds for natural problems like MCSP against weak circuit classes. In most interesting cases, we already have such required lower bounds, but for a different, seemingly easier problem. Firstly, we establish that instantiations of magnification for certain variants of MCSP provably refute the existence of natural proofs against P/poly, which indicates that the natural proofs barrier by Razborov-Rudich may not be relevant to this approach. This is done by establishing a connection from hardness magnification to hardness of efficiently learning polynomial-sized circuits. Next, we investigate the future prospects of proving new lower bounds using magnification. We identify a new source of difficulty called the *locality barrier*, when trying to adapt existing lower bound techniques to prove strong separations via magnification. We show that this barrier applies to many current instantiations of hardness magnification, that involve a wide range of restricted computational models.

Next we consider learnability, and initiate a *top-down* study of hardness of learning circuit classes, i.e. we consider hardness of learning circuit classes more powerful than P/poly. Our goal is to understand the power and limitations of current complexity theoretic techniques in showing *unconditional* results for learning. We show unconditional hardness results for efficiently learning classes like $\mathsf{BPE/poly}, \mathsf{EXP/poly}$ and PSPACE/poly on one hand, and

on the other, show that it is implausible that one can extend these ideas to prove the NP-Hardness of learning classes like non-deterministic polynomial-sized circuits, i.e. NP/poly.

Finally, we consider the algorithmic question of counting the number of satisfying assignments of an input $\mathsf{AC}^0[2]$-circuit. We construct a *deterministic* #SAT-algorithm for $\mathsf{AC}^0[2]$ circuits of depth d and size $2^{O(n^{1/d})}$, which beats brute-force running time non-trivially. The algorithm is designed using the Razborov-Smolensky polynomial method, whose original motivation was to prove $\mathsf{AC}^0[2]$ lower bounds. We then show a lower bound for E^{NP} against $\mathsf{AC}^0[2]$-circuits of size $2^{\Omega(n^{1/d+1})}$, which was the best known lower bound against this class until very recently.

Contents

Chapter 1

Introduction

One of the main roles of complexity theory is to formally study the power and limitations of various models designed to understand computation in an abstract sense, like Turing machines or Boolean circuits. In particular, studying the limits, or *lower bounds* of such computational models is an important endeavour. This is done by associating natural complexity measures (like running time, space, circuit size/depth, number of queries or the amount of randomness) for these models and then proving that certain problems cannot be computed within a model when its associated complexity measures are constrained. Proving a lower bound provides us with a deeper insight into the nature of computation; it says something new about the computational model and thus, algorithms in general. From a more practical viewpoint, complexity lower bounds create boundaries for the efficiency of problems and save us from superfluous attempts to come up with solutions for these problems beyond these boundaries.

Of particular importance is the question of lower bounds against *non-uniform* models of computation, i.e. models where different algorithms can be used to solve the problem on different input lengths (unlike uniform models like Turing machines where one algorithm solves the problem on all input lengths). Non-uniform models are generally represented as Boolean circuits – computational devices obtained by forming a *directed acyclic graph* of logical gates connected to each other by wires. Typically, the gates compute elementary functions like $\mathsf{AND}, \mathsf{OR}$ and NOT. In a very natural way, non-uniform models represent computations which can be performed by hardware. These circuit models are further refined by introducing structural constraints on them like the number of gates, depth of the circuit or types of the gates allowed.

Questions about the power of Boolean circuits are more combinatorial in nature and thus, are more amenable to be tackled by different mathematical techniques. Furthermore, lower bounds on Boolean circuit models also resolve questions about uniform computation, such as the P vs NP problem, or the BPP vs P problem, in addition to answering fundamental questions in pseudorandomness, cryptography and learning theory.

An interesting class of problems to consider are *meta-computational problems*, which *themselves encode questions about circuits or algorithms.* As we shall see, progress in complexity theory is very intimately connected to studying these questions. Typically, the inputs to these problems are Boolean functions, represented using some computational object like a Turing machine/Boolean circuit, or as a truth table. Some famous examples of such problems include the *Halting Problem, Circuit Satisfiability*, or the problem of finding the *Kolmogorov complexity* of an input string. Furthermore, algorithmic solutions for problems like Satisfiability have also found widespread use in practice.

In this thesis, we study computational questions on meta-computational problems. This includes designing algorithms and exploring computational limits of such problems. As part of this process, we also make contributions to the ongoing attempts of establishing connections between the complexity of these problems and lower bounds.

1.1 Meta-Computational Problems

To define some meta-computational problems formally, we first need to fix a circuit class $\mathcal{C}$, which is a set of functions that can be computed by a sequence of Boolean circuits of a certain kind, one for each input length, with refinements obtained by placing constraints on their structure and size (note that the size is measured as a function of the number of inputs). In this thesis, our main focus is on the following meta-computational problems.

- $\mathcal{C}$-SAT : Given a $\mathcal{C}$-circuit C as input, does there exist an input that is accepted by (or satisfies) C?
- $\mathcal{C}$-$\#\mathsf{SAT}$: Given a $\mathcal{C}$-circuit C as input, how many inputs are accepted by C?
- **$\mathcal{C}$-Minimum Circuit Size Problem ($\mathcal{C}$-$\mathsf{MCSP}[s]$)** : Given the truth table of a Boolean function f and a number s, is there a $\mathcal{C}$-circuit of size at most s which computes f?
- **$\mathcal{C}$-Learning** : Given only query access to a function $f \in \mathcal{C}$, can we output a small-sized circuit which approximates it?

It is worth noting that for each of these problems, although the method in which a function is presented as input may differ, i.e. as a circuit, a truth table or an oracle, the function is guaranteed to be from $\mathcal{C}$.

An intuitive reason for the deep connections between $\mathcal{C}$-circuit lower bounds and $\mathcal{C}$-meta-computational problems is that both tasks need an intricate understanding of *all* $\mathcal{C}$-circuits, either to show that a "hard" function can not be computed by them, or to obtain a better analysis of the algorithmic task.

Other important meta-computational questions pertain to the theory of derandomisation and include, the $\mathcal{C}$-Circuit Acceptance Probability Problem ($\mathcal{C}$-CAPP), pseudorandom generators (PRGs) for $\mathcal{C}$ and hitting set generators (HSGs) for $\mathcal{C}$. Note that the construction of PRGs and HSGs differ from typical meta-computational questions, in the sense that no function is provided as an input. Rather, we broadly consider these as algorithmic questions about $\mathcal{C}$, e.g. can we construct an algorithm whose output appears almost uniformly random to any function in $\mathcal{C}$?

We briefly mention some well known circuit classes that have been studied so far. One of the most commonly agreed upon notions of easiness for non-uniform classes is the class $\mathsf{P/poly}$, which contains functions computable by a family of circuits with polynomially many gates (called its *size* - measured as a function of the number of inputs), one for each input length. Another standard class is NC^1, where each function is computable by a sequence of polynomial-sized De Morgan formulas. A De Morgan formula is a binary tree where each leaf is a literal and each internal node is an AND or OR gate. The formula size is given by the number of leaves in the tree.[1] Furthermore, we also look at constant-depth polynomial-sized circuit classes like $\mathsf{AC}^0, \mathsf{AC}^0[2]$ and ACC^0, which have unbounded fan-in $\mathsf{AND}, \mathsf{OR}$, Mod_2 (in case of $\mathsf{AC}^0[2]$) or Mod_k for every integer k (in case of ACC^0), and NOT gates. We refer to [AB09, Juk12] for more details.

Before stating the main contributions of this thesis, we summarise the myriad connections between meta-computational problems and complexity lower bounds.

Circuit Satisfiability

Circuit Satisfiability (or $\mathcal{C}$-SAT) is the canonical meta-computational problem, whose study (or the study of its variants) has led to major advances in complexity theory like the Cook-Levin theorem [AB09] which showed that $\mathsf{CNF\text{-}SAT}$ is NP-Complete, and the PCP theorem [ALM+98, AS98] which showed that $\mathsf{CNF\text{-}SAT}$ is NP-Hard to approximate (here, $\mathcal{C}$ is the class of functions computable by CNFs of polynomial size). In its most general form, $\mathcal{C} = \mathsf{P/poly}$ and the problem is denoted as $\mathsf{Circuit\text{-}SAT}$.

Designing $\mathcal{C}$-SAT algorithms for various circuit classes $\mathcal{C}$ has been a serious research endeavour on its own, from both theoretical and practical standpoints. However, the last decade has seen the addition of another dimension to the study of SAT algorithms – its connections to circuit lower bounds. It turns out that these connections can be used to show lower bounds which have resisted the use of direct approaches so far. Some forms of these connections have been known for a while now. Indeed, a collapse theorem from [KL80] (attributed to Meyer) can be used to show that if $\mathsf{CNF\text{-}SAT}$ has polynomial time algorithms, then EXP (deterministic exponential time) is not in $\mathsf{P/poly}$. Another

[1] A Boolean formula can be viewed as a Boolean circuit where each gate has fan-in 2 and fan-out 1.

well-known connection is via the "Satisfiability coding lemma", which was established by [PPZ97] to analyse their k-CNF-SAT algorithm. This lemma is then used by them to show lower bounds against AC^0-circuits of depth 3.

In more recent developments, Williams gave a refined version of this connection [Wil13a, Wil14c], where a non-trivial algorithm[2] for $\mathcal{C}[\mathsf{poly}]$-SAT implied that $\mathsf{NEXP} \not\subseteq \mathcal{C}[\mathsf{poly}]$, where $\mathcal{C}[\mathsf{poly}]$ is the class of functions computable by polynomial-sized $\mathcal{C}$-circuits and NEXP is non-deterministic exponential time. Additionally, he used this framework to show that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$, by designing a non-trivial ACC^0-SAT algorithm.[3] Interestingly, all known ACC^0-lower bounds are proved via this framework. Other approaches that have been proposed are also based on the design of new meta-computational algorithms for ACC^0 ([CKK+15]). This framework has been called the *algorithmic paradigm* for proving new circuit lower bounds. These connections have also been termed as *transference theorems* ([Oli13]), as they connect two seemingly different lines of research and show that algorithmic results can be "transferred" to results on circuit lower bounds. For a gentle introduction to these connections, see the surveys [San13, Oli13, Wil14a].

Subsequently, many more circuit lower bounds have been proved using this paradigm, by either strengthening the connection between SAT algorithms and lower bounds [Oli13, Wil13b, Wil14b, Wil16, BSV14, MW18, CW19, VW20] and/or introducing new $\mathcal{C}$-SAT or $\mathcal{C}$-#SAT algorithms [Wil14b, Tam16, ACW16, Wil18]. This approach has also contributed towards proving average-case lower bounds for ACC^0 [COS18, Che19, CR20, CLW20]. For example, using a strengthened connection [MW18] show that non-deterministic quasi-polynomial time cannot be computed by polynomial sized ACC^0-THR circuits, via a #SAT algorithm for ACC^0-THR[poly] from [Wil14b].[4]

On the other hand, the theory of circuit lower bounds has also contributed to advances in the design of $\mathcal{C}$-SAT algorithms. Broadly speaking, the lower bound for a weak circuit class $\mathcal{C}$ (like constant-depth circuits) exposes some combinatorial or algebraic "weakness" (i.e. a structural property) specific to all functions in $\mathcal{C}$. Since the input instances to $\mathcal{C}$-SAT are also $\mathcal{C}$-circuits, having such structure over all the input instances can be exploited for algorithm design. In this vein, many recent SAT and #SAT algorithms for classes like AC^0 [BIS12, IMP12], formulas of sub-cubic size [San10, ST13, CKK+15, CKS16, Che15, Tal15, KRT17], and linear-sized general circuits [CK15, GKST16] were developed by adapting the techniques originally intended for their corresponding lower bounds. This

[2] A $\mathcal{C}$-SAT algorithm is non-trivial if it runs in $2^n/n^{\omega(1)}$ time, i.e. it is "just" better than brute-force.

[3] ACC^0 is the class of functions computed by polynomial-sized constant-depth circuits consisting of unbounded fan-in $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}$ and MOD_m gates (modulus gates), for any positive integer m.

[4] ACC^0-THR is the class of functions computable by ACC^0 circuits, with a layer of weighted threshold gates at the bottom.

template has also been extended to design non-trivial algorithms for variants like MAX-SAT [CS15] (when the input CNFs are sparse), where the goal is to find an assignment that maximises the number of satisfied clauses of the input CNF.

Interestingly, a careful analysis of many of these SAT algorithms also gives us a converse connection to circuit lower bounds. Most works above use the analysis of the respective $\mathcal{C}$-SAT (or $\mathcal{C}$-#SAT) algorithms to show strong average-case lower bounds against $\mathcal{C}$. It must be noted that these connections from lower bounds to SAT algorithms are not generic like Williams' framework and are inspired by the specific lower bound proof for the model in question.

$\mathcal{C}$-Minimum circuit size problem ($\mathcal{C}$-MCSP)

While unconditional circuit lower bounds have been known in many restricted settings (cf. [Juk12]), extending this to stronger models has been fruitless so far. In their iconic paper on the theory of *natural proofs*, Razborov and Rudich [RR97] proposed a conditional explanation to address the reason for this difficulty.

A *natural property* $\mathcal{R}$ is a subset of all Boolean functions, such that there is an efficient algorithm (with respect to the truth table size of f) that can check if $f \in \mathcal{R}$ (*constructivity* condition). In addition, it is required that a non-trivial fraction of all functions belong to $\mathcal{R}$ (*density* condition). [RR97] noted that most known circuit lower bound proofs (until then) for classes $\mathcal{C}$ like AC^0 and $\mathsf{AC}^0[2]$, used the following template: identify a natural property $\mathcal{R}$, such that no functions from $\mathcal{C}$ satisfy $\mathcal{R}$ and for some "explicit" hard function h, show that $h \in \mathcal{R}$ ($\mathcal{R}$ is *useful* against $\mathcal{C}$). For example, the lower bound for Parity against AC^0 [Ajt83, FSS84, Yao85, Hås86], identifies $\mathcal{R}$ as the subset of functions which do not collapse to a constant under a suitable random restriction and observes that, while any AC^0 function does simplify, Parity does not.

[RR97] show that under widely accepted cryptographic assumptions (existence of strong one-way functions), there exists no efficient natural property useful against $\mathsf{P/poly}$. They argued that natural proofs are almost *self-defeating*: any lower bound proof using a natural property provides an efficient algorithm (via constructivity), which would in turn refute stronger lower bounds that are believed to hold. The moral of the story is that, techniques successful for proving lower bounds against weak classes[5] do not help us for classes like $\mathsf{P/poly}$ or NC^1, as they provide such efficient algorithms. Informally speaking, circuit lower bound proofs which use sufficiently constructive combinatorial arguments, and apply not just to a specific function like Parity, but also to a dense subset of all functions, cannot be hoped to extend to stronger classes. [NR04] showed that assuming

[5] We refer to the preliminaries (Chapter 2) for inclusion of the circuit classes.

the existence of certain hard number-theoretic problems, there exists no natural property useful against even depth-4 TC^0-circuits.[6] It is still unknown whether classes like ACC^0 or depth-2 TC^0, which are between depth-4 TC^0 and $\mathsf{AC}^0[2]$, have any natural properties as there is no known cryptographic barrier against this.

This theory indicates that proving stronger circuit lower bounds might be tightly related to studying meta-computational problems which *refer to the computational complexity of strings or truth tables*. This has led to the study of MCSP, which put simply, is the problem of deciding the circuit complexity of an input function. The input to $\mathcal{C}$-MCSP is the truth table $\mathsf{tt}(f)$ of a Boolean function f on n variables and a size parameter $s = s(n)$, and the problem is to determine if f can be computed by $\mathcal{C}$-circuits of size at most s. It is worth noting that [HS17] show that a zero-error average-case version of $\mathcal{C}$-MCSP is *equivalent* to a natural property against $\mathcal{C}$.

For the most general case where $\mathcal{C} = \mathsf{P/poly}$, what do we know about the complexity of MCSP? The short answer is – very little. On one hand, under reasonable cryptographic assumptions, Kabanets-Cai [KC00] show that $\mathsf{MCSP} \notin \mathsf{P}$. To lend further evidence to this, problems like factoring and discrete logarithm which are not believed to be solvable in polynomial time, reduce to MCSP [ABK+06]. Even for restricted circuit classes, unproven explicit circuit lower bounds against $\mathcal{C}$ are necessary for the existence of any sufficiently constructive natural properties against $\mathcal{C}$ [IKW02, OS17, IKV18]. In fact, [Wil16] shows that even properties against $\mathcal{C}$ which are just constructive and useful (but are not necessarily large), are equivalent to proving NEXP lower bounds against $\mathcal{C}$. On the other hand, MCSP belongs to NP, as a non-deterministic Turing machine can guess a circuit of size at most s and check if it computes the input truth table, in time polynomial in its input size 2^n.

One of the most important questions surrounding MCSP is its NP-Completeness (when the size parameter is large). In fact, this question has been open since the discovery of the theory of NP-Completeness and despite numerous works since, we don't even have formal evidence supporting or refuting the NP-Hardness of MCSP. Despite this, there has been some success for special variants of $\mathcal{C}$ [AHM+08, HOS18, Ila20a, ILO20, Ila20b]. For the general case, all we have is evidence that no proof of its NP-Completeness will be found anytime soon. Indeed, [KC00, HP15, MW17, AHK17, SS20] show that the NP-Hardness of MCSP under different kinds of reductions, most of which have been commonly used, would imply lower bounds which are beyond the reach of current techniques. Switching perspectives, one can think of these results as connecting uniform hardness results of a meta-computational problem (NP-hardness in this case) to lower bounds in complexity.

[6] TC^0 is the class of functions computable by Boolean circuits of polynomial size and constant depth with unbounded fan-in AND, OR, NOT and weighted threshold gates.

Moving on to non-uniform hardness, MCSP (for certain parameter ranges) cannot be computed by most standard restricted circuit classes [ABK+06, HS17, GII+19, CKLM19, CJW20, CHMY20, KKL+20]. What makes further analysis of MCSP circuit lower bounds interesting, is the striking phenomenon termed as *hardness magnification* by [OS18a]. They show that in several scenarios, even slightly super-linear (i.e. "barely" non-trivial) lower bounds for some variants of MCSP against circuit classes like formulas or AC^0, imply strong circuit lower bounds.

For example, [OS18a] show that if an approximate variant of MCSP cannot be computed by barely super-linear formulas, then NP is not in NC^1. Optimistically speaking, we know that Parity cannot be computed by sub-quadratic formulas [Hås98], and worst-case $\mathsf{MCSP}[2^{n^{o(1)}}]$ cannot be computed by sub-quadratic formulas [HS17]. If any of these lower bound ideas could be extended to approximate MCSP, then [OS18a] would show that $\mathsf{NP} \not\subseteq \mathsf{NC}^1$. For the pessimist, such results indicate that for some problems, weak lower bounds are harder than previously imagined. If we believe that circuit lower bounds such as $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ are very hard to prove, then hardness magnification suggests there should be other deep reasons why we cannot even prove weak lower bounds for MCSP and other variants.

Another related meta-computational problem which also leads to magnification phenomena, is the minimum time-bounded Kolmogorov complexity problem or MKtP. Kt complexity was first defined by Levin [Lev84] as: For any arbitrary universal Turing machine U, $\mathsf{Kt}(x) = \min\{|D| + \log t \mid U(D) = x \text{ in at most } t \text{ steps}\}$. MKtP takes as input x of length n and a parameter $s \leq n$, and decides if $\mathsf{Kt}(x) \leq s$. This problem is also very well studied from the perspective of meta-complexity and is known to be EXP-Complete under $\mathsf{P/poly}$ reductions [ABK+06]. There are many variants of resource-bounded Kolmogorov complexity which are related to MCSP and MKtP. We refer the reader to the papers [All01, ABK+06, All17] for further explanation of these variants.

In some sense, where the theory of natural proofs creates a dichotomy between strong and weak circuit classes, magnification indicates cases where this can be bridged. Where earlier attempts at proving circuit lower bounds for restricted classes presumably served as a natural starting point towards proving stronger lower bounds, magnification suggests that it is *sufficient to study such lower bounds for certain problems* to prove strong lower bounds.[7] In any case (quoting [OS18a]), studying magnification leads to a win-win situation: either we get strong lower bounds or we gain a better understanding of barriers for proving circuit lower bounds. Put together, this has led to a spurt of magnification results of various kinds over the last couple of years [Oli19, OPS19, CT19,

[7] Certain people have also started to expand MCSP as "Makes Complexity Separations Possible".

MMW19, CMMW19, CJW19, CJW20, CHMY20] (See [Sri03, AK10, LW13, MP20] for some previous ones).

MCSP and MKtP (and their variants) also offer philosophical insight into the age-old question of how random a string is. Indeed, these problems equate a string which has a lot of structure, so much so that it can be generated by an efficient algorithm (non-uniform or uniform), to not being random. In addition to its connections to circuit lower bounds, the study of MCSP and MKtP is also related to other areas in theoretical computer science like cryptography [San20, LP20], worst-case to average-case reductions for NP [Hir18], proof complexity [PS19] and learning theory [CIKK16] to name a few. We refer to the recent survey by [All20] for further details of these connections. Considering its potential impact, MCSP has of late emerged as the "new" canonical meta-computational problem.

Circuit Learnability

Understanding what classes of functions can be efficiently learned has been a major research direction in computational learning theory since efficient learning was formalised by Valiant [Val84].

To make the following discussion clearer, we informally define the model of learning a class $\mathcal{C}$ using *membership queries* over the uniform distribution. In this model, the learner is given oracle access to any target Boolean function from $\mathcal{C}$, and its aim is to produce, with high probability, a hypothesis that *approximates* the target function well on the uniform distribution. Note that, the hypothesis h is not required to be from the same class $\mathcal{C}$. Furthermore, we say that the learner is distribution independent, if the learner outputs a correct hypothesis over any fixed target distribution over the inputs. A related model is to learn using only *random examples*, where queries made to the oracle are also drawn independently at random from the fixed target distribution. Precise definitions of the learning models can be found in Chapter 2.

Like SAT algorithms, lower bound techniques have been useful in designing learning algorithms. Linial et al. [LMN93] designed a quasi-polynomial time learning algorithm for AC^0 using random examples over the uniform distribution, by adapting the technique used to show AC^0 lower bounds (i.e. random restrictions and the switching lemma [Hås86]). [ST17] use lower bound techniques like random restrictions and Nečiporuk's method, to obtain distribution independent, exact (no error), *non-trivial* learning algorithms for many classes including $\mathsf{MAJ}\text{-}\mathsf{AC}^0$ and full-basis formulas of sub-quadratic size.

Recall from the last section that natural properties against $\mathcal{C}$ are examples of meta-computational problems which decide the circuit complexity of input truth tables. While the above learning algorithms for $\mathcal{C}$ adapted the respective lower bound techniques, in

a major development, [CIKK16] showed an important connection between two meta-computational problems: any natural proof against $\mathcal{C}$ ($\mathcal{C}$ satisfies a mild technical condition) implies a learning algorithm for $\mathcal{C}$ using membership queries over the uniform distribution. This provides a rare example of a generic connection from circuit lower bounds to learning algorithms. They used this framework to get a quasi-polynomial time learning algorithm for $\mathsf{AC}^0[2]$ using membership queries over the uniform distribution. In the converse direction, [OS17] show that any non-trivial learners for $\mathcal{C}$ can be used to construct certain kinds of natural properties against $\mathcal{C}$.

A related problem is that of $\mathcal{C}$-*Compression*: Given the truth table of a Boolean function $f \in \mathcal{C}$, print a non-trivial circuit that computes f, in polynomial time in the size of the truth table, i.e. in $2^{O(n)}$ time, print a circuit of size less than $2^n/n$ which computes f.[8] Like SAT and learning algorithms, specific lower bound techniques have been adapted to construct deterministic compression algorithms for classes like AC^0, sub-cubic formulas and sub-quadratic full-basis formulas [CKK+15] and for $\mathsf{AC}^0[2]$ [KS18]. [ST17] also obtain deterministic compression algorithms for many other classes like span programs, by adapting a different lower bound technique by Nečiporuk. [CIKK16] also provide a generic connection from natural proofs against $\mathcal{C}$ to randomised compression algorithms for $\mathcal{C}$, building on their previously stated result.

For any stronger circuit classes like ACC^0, we do not have *any* learning or compression algorithms. One reason for this is that obtaining unproven lower bounds against $\mathcal{C}$ are necessary for the existence of $\mathcal{C}$-learners. Indeed, a sequence of results by [FK09, HH13, KKO13, Vol14, OS17, COS18], showed that deterministic or randomised learning algorithms for $\mathcal{C}$ in various models of learning imply unproven lower bounds for classes like EXP, PSPACE or BPEXP against $\mathcal{C}$. In particular, [OS17] showed an analogue of Williams' connection, where non-trivial learners for $\mathcal{C}[\mathsf{poly}]$ imply lower bounds for BPEXP (two-sided bounded-error probabilistic exponential time) against $\mathcal{C}[\mathsf{poly}]$. In a similar vein, [CKK+15] showed that polynomial time deterministic compression algorithms for $\mathcal{C}[\mathsf{poly}]$ also imply NEXP lower bounds against $\mathcal{C}$.

Stronger classes like $\mathsf{P/poly}$ or even depth-4 TC^0 cannot be efficiently learnt using membership queries, under widely acknowledged cryptographic assumptions [KV94a, NR04]. However, we do not have such hardness results from weaker, complexity theoretic assumptions. Even showing the NP-hardness of PAC-learning $\mathsf{P/poly}$ is still out of reach; [ABX08] rule out certain kinds of black-box reductions from SAT to learning $\mathsf{P/poly}$.[9]

[8] Note that every Boolean function can be computed by a circuit of size $2^n/cn$, for some constant $c > 1$ [Sha49].

[9] They also show that certain other kinds of black-box NP-Hardness reductions would have major cryptographic consequences.

For other meta-computational tasks, well-known connections include the *hardness vs randomness* paradigm [NW94, IW97] which connects circuit lower bounds to pseudo-random generator/hitting set generator constructions, and the non-trivial $\mathcal{C}$-CAPP algorithms to circuit lower bounds framework [Wil13a, CW19, CR20, CLW20].

1.2 Main Contributions and Outline of the Thesis

The thesis is thematically composed of two parts. In Chapters 3, 4 and 5 we study the hardness of meta-computational problems like $\mathcal{C}$-MCSP and $\mathcal{C}$-learning, their interconnections and their implications to complexity lower bounds. Following this, in Chapter 6, we study algorithmic results for $\mathcal{C}$-#SAT obtained from lower bound proofs against $\mathcal{C}$. We also provide conclusions and open problems at the end of each chapter, and end the thesis with concluding remarks in Chapter 7.

Chapter 2: Preliminaries
In this chapter, we establish notation that is used over the thesis. Notation which is chapter specific is stated in appropriate places.

Chapter 3: Hardness Magnification and Meta-Computational Problems
Hardness magnification is a phenomenon which reduces major complexity separations (such as $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$) to proving lower bounds for some natural problem Q against weak circuit models. In the most intriguing cases, the required lower bound is known for problems that appear to be significantly easier than Q, while Q itself is susceptible to lower bounds but these are not yet sufficient for magnification.

In this chapter, we consider the following essential question associated with the program: *Does hardness magnification avoid the natural proofs barrier of Razborov and Rudich* [RR97]*?*

We establish that some instantiations of hardness magnification avoid the natural proofs barrier in the following sense: slightly superlinear-size circuit lower bounds for certain versions of MCSP imply the non-existence of natural proofs. We prove this result by establishing that certain magnification theorems not only imply strong worst-case circuit lower bounds, but also rule out the existence of efficient learning algorithms.

In addition, we observe that this reduction naturally offers a refined connection between a variant of $\mathcal{C}$-MCSP and self-learning algorithms for $\mathcal{C}$.[10] Using this connection,

[10] $\mathcal{C}$ is self-learnable if $\mathcal{C}[s(n]$ can be learnt using $\mathcal{C}$-circuits of size $t(n)$ (typically, $2^n/n^{\omega(1)} \geq t(n) \geq s(n)$), such that the learner outputs descriptions of $\mathcal{C}$-circuits as hypotheses which can be interpreted and efficiently evaluated by $\mathcal{C}$-circuits (cf. [OS18b]). Contrast this with proper learning where the learner is also expected to output $\mathcal{C}$-hypotheses, but the learner can belong to *any* circuit class.

we prove that an approximate variant of (THR-AC^0)-MCSP for suitable size parameters, *is computable* by super-linear TC^0 circuits. This is obtained by showing that the learning algorithm for THR-AC^0 by [GS10] is in fact a self-learner.

Chapter 4: The Locality Barrier for Hardness Magnification

In this chapter, we continue our study of hardness magnification and reflect on the potential of using current lower bound techniques to establish strong lower bounds via this program. To this end, we formulate scenarios which make an instantiation of magnification interesting; a situation where we are at the cusp of proving strong lower bounds using magnification. We call such scenarios as *HM Frontiers*. We establish these HM frontiers via a mix of existing and new magnification theorems, and circuit lower bounds.

We then study another important question associated with the hardness magnification program: *Can we adapt known lower bound techniques to establish the desired lower bound for Q?* Continuing from Chapter 3, we observe that hardness magnification might sidestep natural proofs, but we identify a source of difficulty when trying to adapt existing lower bound techniques to prove strong lower bounds via magnification.

This is captured by a *locality barrier*: existing magnification theorems *unconditionally* show that the problems Q considered above admit highly efficient circuits extended with small fan-in oracle gates, while lower bound techniques against weak circuit models quite often easily extend to circuits containing such oracles. This explains why direct adaptations of certain lower bounds are unlikely to yield strong complexity separations via hardness magnification.

Chapter 5: Structure of Learnability beyond $\mathsf{P/poly}$

We now turn our attention to studying lower bounds for the complexity of $\mathcal{C}$-learning. Motivated by the goal of showing stronger structural results about the complexity of learning, we study the learnability of strong concept classes beyond $\mathsf{P/poly}$, such as $\mathsf{PSPACE/poly}$ and $\mathsf{EXP/poly}$. We show the following:

1. *(Unconditional Lower Bounds for Learning)* Building on [KKO13], we prove unconditionally that $\mathsf{BPE/poly}$ cannot be weakly learned in polynomial time over the uniform distribution, even with membership and equivalence queries.

2. *(Robustness of Learning)* For the concept classes $\mathsf{EXP/poly}$ and $\mathsf{PSPACE/poly}$, we show unconditionally that worst-case and average-case learning[11] are equivalent, that PAC-learnability and learnability over the uniform distribution are equivalent, and that membership queries do not help in either case.

[11] Informally, an average-case $\mathcal{C}$-learner outputs a good hypothesis (say, over the uniform distribution on the inputs) for any target function sampled with respect to some fixed distribution over $\mathcal{C}$.

3. *(Reducing Succinct Search to Decision for Learning)* Let $\mathsf{R_{Kt}}$ (respectively $\mathsf{R_{KS}}$) be the set of strings with high Kt complexity (respectively space-bounded Kolmogorov complexity, KS complexity). For the decision problems $\mathsf{R_{Kt}}$ and $\mathsf{R_{KS}}$ capturing the complexity of learning $\mathsf{EXP/poly}$ and $\mathsf{PSPACE/poly}$ respectively, we show a *succinct search to decision* reduction: for each of these problems, the problem is in BPP iff there is a probabilistic polynomial-time algorithm computing circuits which encode proofs for positive instances of the problem. This is shown via a more general result giving succinct search to decision results for $\mathsf{PSPACE}, \mathsf{EXP}$ and NEXP, which might be of independent interest.

4. *(Implausibility of Oblivious Strongly Black-Box Reductions showing* NP*-hardness of learning* $\mathsf{NP/poly}$*)* We define a natural notion of hardness of learning with respect to oblivious strongly black-box reductions. We show that learning $\mathsf{PSPACE/poly}$ is PSPACE-hard with respect to oblivious strongly black-box reductions. On the other hand, if learning $\mathsf{NP/poly}$ is NP-hard with respect to oblivious strongly black-box reductions, the Polynomial Hierarchy collapses.

Chapter 6: Deterministic #SAT Algorithm for $\mathsf{AC^0}[2]$

The previous chapters have studied (uniform and non-uniform) hardness of problems like $\mathcal{C}$-MCSP and $\mathcal{C}$-learning, and their implications to lower bounds. In this chapter, we take a dual approach and study algorithmic upper bounds for $\mathcal{C}$-$\#\mathsf{SAT}$ algorithms, using $\mathcal{C}$-lower bound methods.

Our main result of this chapter is a deterministic algorithm for counting the number of satisfying assignments of any $\mathsf{AC^0}[2]$ circuit C of size s and depth d over n variables in time $2^{n-f(n,s,d)}$, where $f(n,s,d) = n/O(\log(s))^{d-1}$, whenever $s = 2^{o(n^{1/d})}$. The algorithm falls within the general framework of using the *polynomial method* in circuit complexity [Raz87, Smo87] for algorithm design, particularly $\mathcal{C}$-SAT algorithms [Wil14c, Wil14b, CW16, ACW16, LPT+17]. Our algorithm is obtained via a derandomisation of Razborov and Smolensky's probabilistic polynomial construction for $\mathsf{AC^0}[2]$-circuits, which was initially used by them to show $\mathsf{AC^0}[2]$ lower bounds.

As a consequence, we get that for each d, there is a language in $\mathsf{E^{NP}}$ that does not have $\mathsf{AC^0}[2]$ circuits of size $2^{o(n^{1/(d+1)})}$. This is the first lower bound in $\mathsf{E^{NP}}$ against $\mathsf{AC^0}[\oplus]$ circuits that beats the lower bound of $2^{\Omega(n^{1/2(d-1)})}$ due to Razborov and Smolensky for large d. Both our algorithm and our lower bounds extend to $\mathsf{AC^0}[p]$ circuits for any prime p.

1.3 List of Papers

The majority of results in this thesis are from the following series of papers:

- Ninad Rajgopal, Rahul Santhanam, and Srikanth Srinivasan
  Deterministically Counting Satisfying Assignments for Constant-Depth Circuits with Parity Gates, with Implications for Lower Bounds [RSS18]
  Proceedings of *43rd International Symposium on Mathematical Foundations of Computer Science, (MFCS 2018)*.
  The results from this paper appear in Chapter 6.

- Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam
  Beyond Natural Proofs: Hardness Magnification and Locality [CHO+20]
  Proceedings of *11th Innovations in Theoretical Computer Science Conference, (ITCS 2020)*.
  To Appear in *Journal of the ACM (JACM)*.
  Most results in Chapters 3 and 4 appear from this paper.

- Ninad Rajgopal, and Rahul Santhanam
  On the Structure of Learnability Beyond P/poly [RS21]
  Proceedings of *Approximation, Randomization, and Combinatorial Optimization, (APPROX/RANDOM 2021)*.
  The results from this paper appear in Chapter 5.

Chapter 2

Preliminaries

2.1 Boolean Function Complexity - Basics

Let $\mathcal{F} = \{\mathcal{F}_n\}$, where $\mathcal{F}_n$ is the set of all Boolean functions $f_n : \{0,1\}^n \to \{0,1\}$. Any function f defines a corresponding language $L \subseteq \{0,1\}^*$, where for each $n \geq 1$, $L_n = \{0,1\}^n \cap L = f_n^{-1}(1)$.

For any $f_n \in \mathcal{F}_n$, define $\mathsf{tt}(f_n)$ as its truth table of length 2^n. On the other hand, given a string $x \in \{0,1\}^{2^n}$, define $\mathsf{fn}(x)$ as the function on n inputs whose truth table is x. Throughout this thesis, we use $N = 2^n$ as the length of the truth table of any function in $\mathcal{F}_n$.

Let $\mathcal{C} = \{\mathcal{C}_n\}$ be a class of Boolean functions, where each $\mathcal{C}_n \subseteq \mathcal{F}_n$. In this thesis, we are mostly concerned with classes $\mathcal{C}$ computable by Boolean circuits and its variants. For any circuit class $\mathcal{C}$, we say that any function $f \in \mathcal{C}[s(n)]$, if there exists a sequence of $s(n)$-sized (according to some predefined measure for size) $\mathcal{C}$-circuits $\{C_n\}_{n\in\mathbb{N}}$, one for each input length, such that, for every n large enough, C_n computes f_n. Moreover, any class $\mathcal{D}$ is contained in $\mathcal{C}[\mathsf{poly}(n)]$ if for every $f \in \mathcal{D}$, there exists $k \geq 1$ such that $f \in \mathcal{C}[n^k]$.

In particular, $\mathsf{SIZE}[s]$ denotes functions computable by Boolean circuits with AND,OR and NOT gates of fan-in 2 and size at most s, where the number of gates is defined as the size. $\mathsf{SIZE}[\mathsf{poly}(n)]$ is defined as $\mathsf{P/poly}$. Similarly, $\mathsf{Formula}[s]$ denotes functions computable by Boolean formulas over the De Morgan basis (ANDs and ORs) of size at most s, where input leaves are labelled by literals or constants, and gates have fan-in two and fan-out is restricted to be one. Here, we count the number of leaves of a formula as its size.

We also consider standard constant-depth circuit classes like $\mathsf{AC}^0, \mathsf{AC}^0[p], \mathsf{ACC}^0$ and TC^0. Wherever it is clear by context, we will also use the notation $\mathcal{C}_d$ to refer to circuit classes of constant depth d. We also consider circuit classes of logarithmic depth like

NC^1. Note that, $\mathsf{Formula}[\mathsf{poly}(n)]$ is equivalent to NC^1. For size $\mathsf{poly}(n)$, the inclusion of these circuit classes is given by -

$$\mathsf{CNF}, \mathsf{DNF} \subsetneq \mathsf{AC}^0 \subsetneq \mathsf{AC}^0[p] \subsetneq \mathsf{ACC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{P/poly}$$

We use a few other standard notions and complexity classes. We refer to textbooks in computational complexity and circuit complexity [AB09, Juk12] for more details.

Given a sequence of Boolean functions $\{\mathcal{O}_m\}$ with $\mathcal{O}_m : \{0,1\}^m \to \{0,1\}$, we let $\mathcal{C}^{\mathcal{O}}$ denote the extension of $\mathcal{C}$ that consists of functions computable by a sequence $\{C_n\}$ of oracle $\mathcal{C}$-circuits, which have access to oracle gates $\mathcal{O}_m$, for some m.

For every $n \in \mathbb{N}$, define U_n as the uniform distribution over $\{0,1\}^n$. For any distribution W, we use $w \sim W$ to denote an element sampled according to W. A function $f : \{0,1\}^n \to \{0,1\}$ is γ-approximated by a circuit C on n inputs, if $\Pr_{x \sim U_n}[C(x) = f(x)] \geq \gamma$. Further, we say that for any f, g over n inputs, f is γ-approximated by g, if $\Pr_{x \sim U_n}\{f(x) = g(x)\} \geq \gamma$.

2.2 Learnability

We first define the notion of PAC-learning using random examples, introduced in [Val84]. Let $\mathcal{C} = \{\mathcal{C}_n\}$ be a class of functions and $\mathcal{D} = \{\mathcal{D}_n\}$ be a distribution family over $\{0,1\}^*$, where $\mathcal{D}_n$ is a distribution over $\{0,1\}^n$. We call ε as the error parameter and δ as the confidence parameter.

Definition 2.1 (**Worst-case PAC-learning using random examples**). *For any $0 \leq \varepsilon, \delta < 1/2$, a class $\mathcal{C}$ is (ε, δ)-PAC-learnable in the worst-case using random examples in time $T(n)$, if there exists a randomised algorithm $\mathcal{A}$ such that*

- *For every $n \in N$, for every $f \in \mathcal{C}_n$, for every $\mathcal{D}_n$ over $\{0,1\}^n$, $\mathcal{A}$ takes as inputs $1^n, \varepsilon, \delta$, a set of $m = m(n)$ labeled samples $(x_1, f(x_1)), \ldots, (x_m, f(x_m))$ where each $x_i \sim \mathcal{D}_n$, and $w \in \{0,1\}^{r(n)}$ as internal randomness. $\mathcal{A}$ then outputs the description of a circuit h such that*

$$\Pr_{w \in \{0,1\}^{r(n)}, x_1, \ldots, x_m \sim \mathcal{D}_n} \left\{ \Pr_{y \in \mathcal{D}_n} \{h(y) = f(y)\} \geq 1 - \varepsilon \right\} \geq 1 - \delta$$

- *$\mathcal{A}$ runs in time at most $T(n)$.*[1]

We can also restrict the learnability to a fixed distribution like the uniform distribution U_n, where the learner takes random examples chosen over U_n and hypothesis error is also measured over U_n.

[1]Note that this immediately implies that $m(n) \leq T(n)$

Furthermore, we can extend this definition to PAC-learning over membership queries by giving $\mathcal{A}$ oracle access to the target function $f \in \mathcal{C}_n$, in addition to the random examples drawn from some fixed distribution $\mathcal{D}_n$.

We also find it useful to define $\mathcal{C}$-learning, where the learning algorithm is now *non-uniform*.

Definition 2.2 (Non-uniform learners using membership queries)**.** *For any $0 \leq \varepsilon, \delta < 1/2$ and size function $t : \mathbb{N} \to \mathbb{N}$, a circuit class $\mathcal{C}$ is (ε, δ)-learnable over the uniform distribution by $\mathcal{D}[t(n)]$-circuits, if there exists a sequence of randomised oracle $\mathcal{D}$-circuits $\{\mathcal{A}^{(\cdot)_n}\}_{n\in\mathbb{N}}$, such that for every $f : \{0,1\}^n \to \{0,1\}$ computable by a $\mathcal{C}$-circuit, when $\mathcal{A}^{(\cdot)_n}$ is given oracle access to f and an input random string $w \in \{0,1\}^{t(n)}$, $\mathcal{A}^{f_n}(w)$ outputs the description of a circuit such that*

$$\Pr_w\{\mathcal{A}^{f_n}(w) \text{ } (1-\varepsilon)\text{-approximates } f\} \geq 1 - \delta.$$

$\mathcal{A}^{(\cdot)_n}$ uses non-adaptive membership queries if the set of queries which $\mathcal{A}^{(\cdot)_n}$ makes to the oracle does not depend on the answers to previous queries.

We refer to [KV94b] for more details on PAC-learning.

2.3 Minimum Circuit Size Problem (MCSP)

Recall that $N = 2^n$. For any $n \leq s(n) \leq 2^n/n$, the minimum circuit size problem with parameter $s(n)$ or $\mathsf{MCSP}[s(n)]$, takes a truth table $w \in \{0,1\}^N$ as input and accepts it iff there exists a circuit of size $s(n)$ which computes $\mathsf{fn}(w)$. For any circuit class $\mathcal{C}_d$, consider a natural variant of this problem $\mathcal{C}_d$-$\mathsf{MCSP}[s(n)]$ which checks if the input string can be compressed to a $\mathcal{C}_d$-circuit of size at most $s(n)$.

Definition 2.3 (Generalisation of MCSP)**.** *Let $s, t : \mathbb{N} \to \mathbb{N}$, where $s(n) \leq t(n)$ and $0 \leq \varepsilon, \sigma < 1/2$. Define $\mathsf{MCSP}[(s,\sigma),(t,\varepsilon)]$ on inputs of length $N = 2^n$, as the following promise problem:*

- *YES instances: $y \in \{0,1\}^N$ such that there exists a circuit of size $s(n)$ that $(1-\sigma)$-approximates f_y.*
- *NO instances: $y \in \{0,1\}^N$ such that no circuit of size $t(n)$ $(1-\varepsilon)$-approximates f_y.*

We refer to $\mathsf{MCSP}[(s,0),(t,0)]$ as $\mathsf{MCSP}[s,t]$. Informally, if $\varepsilon > 0$, we say that $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ is an *approximate* version of MCSP. If $\varepsilon = 0$, we refer to it as the *worst-case* version of MCSP or just *Gap* MCSP (when $s < t$).

Remark 2.4. *In Definition 2.3, if $s(n) = t(n)$, we also require that $\sigma \leq \varepsilon$ for the yes and no instances to be disjoint.*

2.4 Kolmogorov Complexity

Fix a universal Turing machine U. Levin [Lev84] defined the following notion of time-bounded Kolmogorov complexity: The Kt complexity of a string x or $\mathsf{Kt}(x)$, is the minimum over $|p| + \log(t)$, where p is a program and $|p|$ is its description length, such that $U(p) = x$ in at most t steps. It is known from [All01] that $\mathsf{Kt}(x)$ is polynomially related to the size of the smallest EXP-oracle circuit computing the function with truth table x (truncating x to its longest initial segment with length a power of two).

The problem $\mathsf{MKtP}[s]$ takes a string $x \in \{0,1\}^N$ as input and accepts it iff $\mathsf{Kt}(x) \leq s$. We also define the promise version of this problem $\mathsf{MKtP}[s,t]$, called as *Gap* MKtP, where the YES instances are those with Kt complexity at most s and NO instances are those with Kt complexity larger than t (cf. [OPS19]).

Similarly, $\mathsf{KS}(x)$ is the minimum over $|p| + s$ such that $U(p) = x$ in at most space s. It is known from [All01] that $\mathsf{KS}(x)$ is polynomially related to the size of the smallest PSPACE-oracle circuit computing the function with truth table x (truncating x to its longest initial segment with length a power of two).

Define $\mathsf{R_{Kt}}$ as the language consisting of strings x such that $\mathsf{Kt}(x) \geq |x|/2$ [ABK+06]. Similarly, define $\mathsf{R_{KS}}$ as the language consisting of strings x such that $\mathsf{KS}(x) \geq |x|/2$ [ABK+06].

2.5 Fourier Expansion

For convenience, we use the $\{-1,1\}$ realisation of the Boolean domain (i.e. -1 represents True and 1 represents False) in this section. We also assume that the output of the Boolean function is a real value.

Any function $f : \{-1,1\}^n \to \mathbb{R}$ has a *unique* multilinear polynomial representation:

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \cdot \chi_S(x)$$

where the monomial $\chi_S(x) = \prod_{i \in S} x_i$ is the parity of x over the subset S, and the Fourier coefficients $\widehat{f}(S) \in \mathbb{R}$ are given by

$$\widehat{f}(S) = \mathbf{E}_x[f(x) \cdot \chi_S(x)]$$

In particular, any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ can be represented by a multilinear polynomial of degree at most n where each coefficient is between $[-1,1]$.

The *Fourier weight* of $f : \{-1,1\}^n \to \mathbb{R}$ on a set S is defined to be $\widehat{f}(S)^2$. For any function $f : \{-1,1\}^n \to \mathbb{R}$, we say that the Fourier weight of f is $(1-\varepsilon)$-concentrated on degree t, if $\sum_{S \subseteq [n], |S| \leq t} \widehat{f}(S)^2 \geq 1 - \varepsilon$. See [O'D14] for more details about the Fourier expansion and proofs.

2.6 Concentration Inequalities

We use the following concentration bounds in the thesis.

Lemma 2.5 (Markov's inequality)**.** *If X is a non-negative random variable and $a > 0$, then*

$$\mathsf{Pr}\{X \geq a\} \leq \frac{\mathbf{E}[X]}{a}$$

Another concentration inequality we use is Chebyshev's inequality, which is a consequence of Markov's inequality.

Lemma 2.6 (Chebyshev's inequality)**.** *Let X be a random variable. Then, for any $t > 0$,*

$$\mathsf{Pr}\{|X - \mathbf{E}[X]| > t\} \leq \frac{\mathsf{Var}(X)}{t^2}$$

where $\mathsf{Var}(X)$ is the variance of X.

Finally, we make use of Hoeffding's inequality which provides a stronger tail bound.

Lemma 2.7 (Hoeffding's inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $0 \leq X_i \leq 1$ for every $i \in [n]$. Let $X = \sum_{i=1}^{n} X_i$. Then, for any $\varepsilon > 0$, we have*

$$\mathsf{Pr}\{|X - \boldsymbol{E}[X]| \geq t\} \leq 2\exp(-2t^2/n)$$

As an application of Lemma 2.7, we show the existence of functions which cannot be approximated by polynomial-sized circuits.

Lemma 2.8 (cf. Lemma 4 in [OS17])**.** *For any $s(n) \geq n$ and $\delta \in [0, 1/2]$, we have*

$$\Pr_{f \sim \mathcal{F}_n}\{\exists \text{ circuit of size } \leq s(n) \text{ computing } f \text{ on } \geq (1/2+\delta)\text{-fraction of the inputs}\} \leq \exp(-\delta^2 2^n + 10s\log s)$$

Chapter 3

Hardness Magnification and Meta-computational Problems

3.1 Introduction

Proving circuit size lower bounds for explicit Boolean functions is a central problem in complexity theory. Unfortunately, it is also notoriously hard, and arguments ruling out a wide range of approaches have been discovered. The most prominent of them is the *natural proofs barrier* of Razborov and Rudich [RR97].

A candidate approach for overcoming this barrier was investigated recently by Oliveira and Santhanam [OS18a]. *Hardness Magnification* identifies situations where strong circuit lower bounds for explicit Boolean functions (e.g. $\mathsf{NP} \not\subseteq \mathsf{P/poly}$) follow from much weaker (e.g. slightly superlinear) lower bounds for specific natural problems. As discussed in [OS18a] (and Section 1.1), in some cases the lower bounds required for magnification are already known for explicit problems, but not yet for the problem for which the magnification theorem holds. This approach to lower bounds has attracted the interest of several researchers, and a number of recent works have proved various magnification results. We provide a concise review of existing results next, to put magnification in perspective.

3.1.1 Previous Work

We focus on some representative examples. For definitions and more details, we refer to Section 2 or the original papers.

Srinivasan [Sri03] (Informal). If there exists $\varepsilon > 0$ such that $n^{1-o(1)}$-approximating MAX-CLIQUE requires Boolean circuits of size at least $m^{1+\varepsilon}$ (where $m = \Theta(n^2)$), then $\mathsf{NP} \not\subseteq \mathsf{SIZE[poly]}$.

Allender-Koucký [AK10] and Chen-Tell [CT19]. Let BFE be the *Boolean formula evaluation problem*, $\mathsf{W}_{\mathsf{S}_5}$ be the *monoid problem on symmetric group* S_5 and $\mathsf{W5\text{-}STCONN}$ be the *s-t connectivity problem on directed graphs of width* 5.
The following results hold.

- Let $\Pi \in \{\mathsf{BFE}, \mathsf{W}_{\mathsf{S}_5}, \mathsf{W5\text{-}STCONN}\}$. Suppose that for each $c > 1$, there exist infinitely many $d \in \mathbb{N}$, such that TC^0 circuits of depth d require more than $n^{1+c^{-d}}$ wires to solve Π. Then, $\mathsf{NC}^1 \not\subseteq \mathsf{TC}^0$.

- Suppose that for each $c > 1$, there exist infinitely many $d \in \mathbb{N}$, such that $\mathsf{Majority}$ cannot be computed by ACC^0 circuits of depth d with $n^{1+c^{-d}}$ wires. Then $\mathsf{Majority} \notin \mathsf{ACC}^0$, and consequently $\mathsf{TC}^0 \not\subseteq \mathsf{ACC}^0$.

Lipton-Williams [LW13]. Let $\mathsf{CircEval}$ be the *Circuit Evaluation Problem*. If there is $\varepsilon > 0$, such that for every $\delta > 0$ we have $\mathsf{CircEval} \notin \mathsf{Size\text{-}Depth}[n^{1+\varepsilon}, n^{1-\delta}]$, then for every $k \geq 1$ and $\gamma > 0$ we have $\mathsf{CircEval} \notin \mathsf{Size\text{-}Depth}[n^k, n^{1-\gamma}]$ (in particular $\mathsf{P} \not\subseteq \mathsf{NC}^1$).

Oliveira-Santhanam [OS18a]. The following results hold.

- Let $s(n) = n^k$ and $\delta(n) = n^{-k}$, where $k \in \mathbb{N}$. If $\mathsf{MCSP}[(s,0),(s,\delta)] \notin \mathsf{Formula}[N^{1+\varepsilon}]$ for some $\varepsilon > 0$, then there is $L \in \mathsf{NP}$ over m-bit inputs, where $m = \mathsf{poly}(s)$, and $\gamma > 0$ such that $L \notin \mathsf{Formula}[2^{m^\gamma}]$.

- Suppose there exists $k \geq 1$, such that for every $d \geq 1$ there is $\varepsilon_d > 0$ for which $\mathsf{MCSP}[(s,0),(s,\delta)] \notin \mathsf{AC}^0_d[N^{1+\varepsilon_d}]$, where $s(n) = n^k$ and $\delta(n) = n^{-k}$. Then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

- Let $k(n) = n^{o(1)}$. If there exists $\varepsilon > 0$ such that $k\text{-}\mathsf{Vertex\text{-}Cover} \notin \mathsf{DTISP}[m^{1+\varepsilon}, m^{o(1)}]$ (deterministic time-space), where the input is an n-vertex graph represented by an adjacency matrix of bit length $m = \Theta(n^2)$, then $\mathsf{P} \neq \mathsf{NP}$.

- Let $k(n) = (\log n)^C$, where $C \in \mathbb{N}$ is arbitrary. If for every $d \geq 1$ there exists $\varepsilon > 0$ such that $k\text{-}\mathsf{Vertex\text{-}Cover} \notin \mathsf{AC}^0_d[m^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

Oliveira-Pich-Santhanam [OPS19] and McKay-Murray-Williams [MMW19] (Informal). If there exists $\varepsilon > 0$ such that for every small enough $\beta > 0$,

- $\mathsf{MCSP}[2^{\beta n}] \notin \mathsf{SIZE}[N^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{P/poly}$.

- $\mathsf{MKtP}[2^{\beta n}] \notin \mathsf{TC}^0[N^{1+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{TC}^0[\mathsf{poly}]$.

- $\mathsf{MKtP}[2^{\beta n}] \notin U_2\text{-}\mathsf{Formula}[N^{3+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$.

- $\mathsf{MKtP}[2^{\beta n}] \notin B_2\text{-}\mathsf{Formula}[N^{2+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$.
- $\mathsf{MKtP}[2^{\beta n}] \notin \mathsf{Formula\text{-}XOR}[N^{1+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$.
- $\mathsf{MKtP}[2^{\beta n}] \notin \mathsf{BP}[N^{2+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{BP}[\mathsf{poly}]$.
- $\mathsf{MKtP}[2^{\beta n}] \notin (\mathsf{AC}^0[6])[N^{1+\varepsilon}]$, then $\mathsf{EXP} \not\subseteq \mathsf{AC}^0[6]$.

Many magnification results for MKtP admit analogues for MrKtP, introduced by [Oli19], which considers a randomised version of Kt complexity. An advantage of MrKtP is that strong unconditional lower bounds against uniform computations are known, while such hardness for problems such as MCSP and MKtP currently relies on cryptographic assumptions.

[MMW19] also show magnification from weak lower bounds against one-pass streaming algorithms to separating P and NP, which was later observed to extend to weak lower bounds against one-tape Turing machines by [CHMY20].

Chen-McKay-Murray-Williams [CMMW19] and Chen-Jin-Williams [CJW19] (Informal). The following results hold.

- If there is $\varepsilon > 0$, $c \geq 1$, and an n^c-sparse language $L \in \mathsf{NP}$ such that $L \notin \mathsf{SIZE}[n^{1+\varepsilon}]$, then $\mathsf{NE} \not\subseteq \mathsf{SIZE}[2^{\delta \cdot n}]$ for some $\delta > 0$.
- If there is $\varepsilon > 0$, such that for every $\beta \in (0, 1)$, there exists a 2^{n^β}-sparse language $L_\beta \in \mathsf{NP}$, where $L_\beta \notin \mathsf{SIZE}[n^{1+\varepsilon}]$, then for every $k > 0$, $\mathsf{NP} \not\subseteq \mathsf{SIZE}[n^k]$.

[CJW19] established that many hardness magnification theorems for problems such as MCSP hold in fact under the assumption that a *sufficiently sparse and explicit family of languages* in NP admits weak lower bounds in many models previously considered, leading to arbitrary fixed-polynomial NP lower bounds. We refer to their work for more details.

[CJW20] (Informal). If there exists $\varepsilon > 0$ such that for every small enough $\beta > 0$,

- $\mathsf{MCSP}[2^{\beta n}]$ does not have $N^{2+\varepsilon}$-size probabilistic formulas, then $\oplus\mathsf{P} \not\subseteq \mathsf{NC}^1$.
- $\mathsf{MKtP}[2^{\beta n}]$ does not have $N^{2+\varepsilon}$-size probabilistic formulas, then $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$.

Interestingly, [CJW20] also show an *unconditional lower bound* for MCSP and MKtP against $N^{2-\varepsilon}$-sized probabilistic formulas (with the same size parameter). In other words, they show an arbitrarily small gap between the magnification threshold and known lower bounds for probabilistic formulas, which is known for very few other magnification results.

Hardness Magnification results have also been established in related areas, such as extended-Frege lower bounds in proof complexity [MP20] and non-commutative arithmetic circuit lower bounds [CILM18].

3.1.2 Our Results

The theory of natural proofs [RR97] suggests that proving strong unconditional circuit lower bounds might be tightly related to the study of meta-computational problems which decide the computational complexity of strings or truth tables. The theory of hardness magnification provides further motivation to study such problems, and in fact, builds off the natural proofs barrier, which indicates the hardness of MCSP, to attack strong circuit lower bounds.

A glance at the various magnification results in Section 3.1.1 suggests a striking phenomenon associated with such problems. For models like De Morgan formulas ([OPS19, MMW19]), if we could establish lower bounds for problems like MCSP or MKtP (in certain parameter ranges) that even marginally improve the current state of art (which is $N^{3-o(1)}$), then super-polynomial *explicit* lower bounds (like $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$) would follow.

More intriguingly, for certain models like Formula-XOR (formulas with a bottom layer of XOR gates), even a super-linear lower bound for MKtP, which is below the best-known, quadratic lower bound against this model (but for another problem Inner-Product), implies that $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$. In another interesting case, [OPS19] show that if $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{SIZE}[N^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{P/poly}$. On the other hand, [HS17] show sub-quadratic *formula* size lower bounds for MCSP in the same parameter range. Thus, if the [OPS19] magnification theorem could be made to work for formulas instead, $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ would follow unconditionally! Such examples inspire further exploration of MCSP lower bounds, as well as associated hardness magnification phenomena. We study these instantiations in more detail in Chapter 4.

As such, most results in this program are still centered on proving weak lower bounds for problems like MCSP or MKtP. Magnification for such meta-complexity problems crucially exploits the difference between their feasible (succinct inputs) and infeasible (truth table inputs) formulations. Notwithstanding this, there are a few examples of magnification from weak lower bounds for other problems like k-Vertex Cover [OS18a], or even the existence of a sufficiently sparse and explicit family of languages in NP [CJW19].[1]

In this chapter, we attempt to expand our intuition on magnification phenomena for MCSP (and variants), by studying the following question: *Is hardness magnification a non-naturalising approach to circuit lower bounds?* If we accept standard cryptographic assumptions, non-naturalisability is a necessary property of any successful approach to strong circuit lower bounds [RR97].

The initial hardness magnification theorem of Oliveira and Santhanam [OS18a] proposes to approach $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ by deriving slightly super-linear circuit lower bounds

[1] For example, $\mathsf{MCSP}[s(n)]$ is a sparse language in NP: On inputs of length N, the language contains at most $2^{O(s(n)\log s(n))}$ truth tables of length N, computable by circuits of size $s(n)$.

for specific problems such as approximate MCSP (with appropriate parameters), which asks to distinguish truth-tables of Boolean functions computable by small circuits from truth-tables of Boolean functions which are hard to approximate by small circuits. Interestingly, this approach does not seem to naturalise, as it appears to yield strong lower bounds only for certain problems, and not for most of them (the same heuristic argument appears in [AK10]). However, this is only an informal argument, and we would like to get stronger evidence that the natural proofs barrier does not apply here.

One way of avoiding the natural proofs barrier is to prove that there are no natural proofs. We show that hardness magnification for approximate MCSP can be used to conclude the *non-existence* of natural proofs against polynomial-size circuits. More precisely, we prove that if approximate MCSP requires slightly super-linear-size circuits, then there are no P/poly-natural properties against P/poly. This suggests that the natural proofs barrier isn't relevant to the magnification approach. Indeed, there remains the possibility that the weak circuit lower bound for approximate MCSP in the hypothesis of the result can be shown using naturalising techniques (as there aren't any strong enough plausible cryptographic conjectures known that rule this out), and yet by using magnification to "break" naturalness, we could get strong circuit lower bounds and even conclude the non-existence of natural proofs!

The core of our proof is the following new hardness magnification theorem: if approximate MCSP requires slightly super-linear-size circuits, then not only $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ but *it is impossible even to learn efficiently*. In other words, we establish a highly efficient reduction between two well-studied meta-computational tasks – from approximate MCSP to efficiently learning P/poly. We can then refute the existence of natural proofs by applying the known translation of natural properties to learning algorithms [CIKK16]. Similar implications hold with a *worst-case gap version* of MCSP instead of approximate MCSP, following an idea from [Hir18].

Interestingly, all the implications above are actually *equivalences*. In particular, the existence of natural properties is equivalent to the existence of highly efficient circuits for computing approximate MCSP and worst-case gap MCSP with certain parameters (cf. Theorem 3.1). This extends a known characterisation of natural properties: [CIKK16] showed that P/poly natural proofs against P/poly are equivalent to learning P/poly by subexponential-size circuits, which was in turn shown to be equivalent by Oliveira and Santhanam [OS17] to the non-existence of non-uniform pseudorandom function families of sub-exponential security. The connection of hardness magnification to learning and pseudorandom function generators might be of independent interest, since it extends the consequences of magnification into two central areas in Complexity Theory.

Theorem 3.1 (Equivalences for Hardness Magnification)**.** *The following statements are equivalent*[2]*:*

(a) ***Hardness of approximate* MCSP *against almost-linear size circuits.*** *There exist* $c \geq 1$*,* $0 < \gamma < 1$*, and* $\varepsilon > 0$ *such that* $\mathsf{MCSP}[(n^c, 0), (2^{n^\gamma}, n^{-c})] \notin \mathsf{SIZE}[N^{1+\varepsilon}]$.

(b) ***Hardness of worst-case* MCSP *against almost-linear size circuits.*** *There exists* $c \geq 1$ *and* $\varepsilon > 0$ *such that* $\mathsf{MCSP}[n^c, 2^n/n^{c-2}] \notin \mathsf{SIZE}[N^{1+\varepsilon}]$.

(c) ***Hardness of sub-exponential size learning using non-adaptive queries.*** *There exist* $\ell \geq 1$ *and* $0 < \gamma < 1$ *such that* $\mathsf{SIZE}[n^\ell]$ *cannot be learned up to error* $O(1/n^\ell)$ *with confidence* $1/n$ *under the uniform distribution, by circuits of size* $2^{O(n^\gamma)}$ *using non-adaptive membership queries.*

(d) ***Non-existence of natural properties against polynomial size circuits.*** *For some* $d \geq 1$ *there is no* $\mathsf{SIZE}[\mathsf{poly}(N)]$*-natural property useful against* $\mathsf{SIZE}[n^d]$.

(e) ***Existence of non-uniform* PRFs *secure against sub-exponential size circuits.***
For every constant $a \geq 0$*, there exists* $d \geq 1$*, a sequence* $\mathfrak{F} = \{\mathcal{F}_n\}_{n\geq 1}$ *of families* $\mathcal{F}_n$ *of* n*-bit Boolean functions* $f_n \in \mathsf{SIZE}[n^d]$*, and a sequence of probability distributions* $\mathfrak{D} = \{\mathcal{D}_n\}_{n\geq 1}$ *supported over* $\mathcal{F}_n$ *such that, for infinitely many values of* n*,* $(\mathcal{F}_n, \mathcal{D}_n)$ *is pseudo-random function family that* $(1/N^{\omega(1)})$*-fools* (*oracle*) *circuits of size* $2^{a\cdot n}$.

The proof of this result appears in Section 3.3. Our main technical contributions in this proof are the connections from Items (a) to Item (c) (and Item (b) to Item (c)). We highlight below the most interesting implications of Theorem 3.1. (Note that some of them have appeared in other works in similar or related forms (like [CIKK16, OS17])).

(a) $\rightarrow$ (d): The initial hardness magnification result from [OS18a, Theorem 1] (stated for circuits) implies the *non-existence* of natural proofs useful against polynomial-size circuits, indicating that the natural proofs barrier might not be relevant to the magnification approach. Additionally, this indicates that magnification instantiations from certain variants of MCSP could be "special", in the sense that they provably refute the existence of natural properties against P/poly.

[2]See Preliminaries (Sections 2 and 3.2) for definitions.

(a), $(b) \leftrightarrow (d)$: *Any* $\mathsf{P/poly}$ natural property useful against $\mathsf{P/poly}$ can be transformed into an almost-linear size natural property which is simply $\mathsf{MCSP}[(n^c, 0), (2^{n^\gamma}, n^{-c})]$ or worst-case gap $\mathsf{MCSP}[n^c, 2^n/n^{c-2}]$. (Note the different regime of circuit size parameters for these problems.)

(a), $(b) \leftrightarrow (c)$: A weak-seeming hardness assumption for worst-case gap and approximate versions of MCSP implies a strong non-learnability result: polynomial-size circuits cannot be learned over the uniform distribution even non-uniformly in sub-exponential time.

(a), $(b) \leftrightarrow (e)$ Hardness magnification for MCSP also yields cryptographic hardness in a certain regime.

We note that the use of non-adaptive membership queries in Theorem 3.1, Item (c) is not essential. It follows from [CIKK16] that, in the context of learnability of polynomial size circuits under the uniform distribution in sub-exponential time, adaptive queries are not significantly more powerful than non-adaptive queries.[3]

Towards a more robust theory of magnification: While Theorem 3.1 formally connects hardness magnification and natural properties, it would be very interesting to understand to what extent different hardness magnification theorems provably refute the existence of natural properties. This would provide a more complete answer to the question on naturalisability asked above.

For instance, Theorem 3.1 leaves open whether hardness magnification for worst-case versions of MCSP such as $\mathsf{MCSP}[n^c, 2^{n^\varepsilon}]$ refutes natural proofs as well. Note that one way of approaching this question would be to study highly efficient reductions from $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$ to its approximate version $\mathsf{MCSP}[(n^{c'}, 0), (2^{n^{\gamma'}}, n^{-c'})]$.[4] In Section 3.4, we observe that this question is strongly related to the problem of basing *hardness of learning* on *worst-case assumptions* such as $\mathsf{P} \neq \mathsf{NP}$ (cf. [ABX08]).

Upper Bounds for $\mathcal{C}$-MCSP: For any circuit class $\mathcal{C}$, define $\mathsf{MAJ\text{-}MAJ\text{-}}\mathcal{C}$ as the circuit class defined by the top two layers being $\mathsf{Majority}$ gates, whose inputs are fed by outputs of $\mathcal{C}$-circuits. Similarly, $\mathsf{THR\text{-}}\mathcal{C}$ is the circuit class $\mathcal{C}$ with a weighted threshold gate as the output gate.

The proof of the connection between hardness magnification for approximate MCSP and hardness of learning in Theorem 3.1 lends itself to the following generalisation: For

[3] In a bit more detail, one can easily extract a natural property from a learner that uses adaptive queries. In turn, closer inspection of the technique of [CIKK16] shows that a non-adaptive learner can be obtained from a natural property.

[4] More precisely, the existence of a reduction from $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$ to $\mathsf{MCSP}[(n^{c'}, 0), (2^{n^{\gamma'}}, n^{-c'})]$ shows that lower bounds for the former problem yield lower bounds for the latter. Since any such lower bound must be non-naturalisable by Theorem 3.1, we obtain the same consequence for $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$ (note that in the context of hardness magnification it is also important to have highly efficient reductions.).

a standard circuit class $\mathcal{C}$, if approximate $\mathcal{C}$-MCSP requires super-linear circuits of the form MAJ-MAJ-$\mathcal{C}$, then there exists no *self-learners* for $\mathcal{C}$, where $\mathcal{C}$ has a self-learner if it can be learnt by non-uniform $\mathcal{C}$-circuits using $\mathcal{C}$-circuits as hypotheses. Self-learners were introduced by [OS18b] in the context of pseudo-deterministic derandomisation of learning algorithms.

We show that approximate THR-AC^0-MCSP has a super-linear TC^0 *upper bound*. This is done by proving that the learning algorithm for THR-AC^0 by [GS10] is actually a self-learner, and using this in the connection stated above. Note that, the observation about self-learning THR-AC^0 was first made by [OS18b] (cf. Section 3.3) and we provide a proof of this statement. Such a result indicates that one must be careful while trying to show a lower bound for approximate $\mathcal{C}$-MCSP towards the goal of magnification, and that the class $\mathcal{C}$ plays an important role in the hardness of $\mathcal{C}$-MCSP.

3.2 Preliminaries

3.2.1 Natural Properties

Definition 3.2 (Natural property [RR97])**.** *Let $\mathfrak{R} = \{\mathcal{R}_n \subseteq \mathcal{F}_n\}_{n\in\mathbb{N}}$ be a combinatorial property of Boolean functions, $\mathcal{C}$ be a circuit class and Γ be a complexity class. Then, $\mathfrak{R}$ is a Γ-natural property useful against $\mathcal{C}[s(n)]$, if there exists an $n_0 \in \mathbb{N}$ such that the following hold:*

- ***Constructivity** : For any function $f_n \in \mathcal{F}_n$, the predicate $f_n \overset{?}{\in} \mathcal{R}_n$ is computable in Γ, in the size of the truth table of f_n.*
- ***Largeness** : For every $n \geq n_0$, $\Pr_{f_n\sim\mathcal{F}_n}\{f_n \in \mathcal{R}_n\} \geq \frac{1}{2^{O(n)}} = \frac{1}{\mathsf{poly}(N)}$.*
- ***Usefulness** : For every $n \geq n_0$, $\mathcal{R}_n \cap \mathcal{C}[s(n)] = \emptyset$.*

The following result which follows from [CIKK16] connects the existence of natural properties useful against a class $\mathcal{C}$ to designing learning algorithms for $\mathcal{C}$.

Theorem 3.3 (From Theorem 5.1 of [CIKK16] and Lemma 14 of [IW01])**.** *Let R be a P/poly-natural property useful against $\mathsf{SIZE}[n^d]$ for some $d \geq 1$. Then, for each $\gamma \in (0, 1)$, there are randomised, oracle circuits $\{D_n\}_{n\geq 1} \in \mathsf{SIZE}[2^{O(n^\gamma)}]$ that learn $\mathsf{SIZE}[n^k]$ up to error $\frac{1}{n^k}$ and confidence $1/n$ using non-adaptive oracle queries to f_n, where $k = \frac{d\gamma}{a}$ and a is a universal constant that does not depend on d and γ.*

3.2.2 PRFs

Definition 3.4 (Pseudorandom function families)**.** *For any circuit class $\mathcal{C}$, size functions $s(n), t(n) \geq n$, family $\mathcal{G}_n$ of n-bit Boolean functions and distribution $\mathcal{D}_n$ over $\mathcal{G}_n$, we say that a pair $(\mathcal{G}_n, \mathcal{D}_n)$ is a $(t(n), \varepsilon(n))$-pseudorandom function family (PRF) in $\mathcal{C}[s(n)]$, if each function in $\mathcal{G}_n$ is in $\mathcal{C}[s(n)]$ and for every randomised circuit $A^O \in \mathsf{SIZE}^O[t(n)]$, where O denotes oracle access to a fixed Boolean function over n inputs, we have*

$$\left| \Pr_{g \sim \mathcal{D}_n, w} \{A^g(w) = 1\} - \Pr_{f \sim \mathcal{F}_n, w} \{A^f(w) = 1\} \right| \leq \varepsilon(n)$$

[OS17] state an equivalence between the non-existence of PRFs in a circuit class $\mathcal{C}$ and learning algorithms for $\mathcal{C}$. In particular, we use the following direction which they prove using a small-support version of Von-Neumann's Min-max Theorem.

Theorem 3.5 (No PRFs in $\mathcal{C}$ implies Learning Algorithm for $\mathcal{C}$ [OS17] (Proposition 35))**.** *Let $t(n) \leq 2^{O(n)}$. Suppose that for every $k \geq 1$ and large enough n, there exists no $(\mathsf{poly}(t(n)), 1/10)$-pseudorandom function families in $\mathcal{C}[n^k]$. Then, for every $\varepsilon > 0, k \geq 1$ and large enough n, there is a randomised oracle circuit family in $\mathsf{SIZE}^O[2^{n^\varepsilon}]$ that learns every function $f_n \in \mathcal{C}[n^k]$ up to error $1/n^k$ with confidence $1/n$, where O denotes membership query access to f_n.*

We refer the reader to Chapter 2 for any other relevant definitions.

3.3 Equivalences in Theorem 3.1

The main contribution of this section is a new hardness magnification result showing non-learnability of circuit classes from slightly super-linear lower bounds for the approximate version of MCSP and the gap version of MCSP. We then use these magnification results to establish a series of equivalences.

Lemma 3.6 (Hardness Magnification for Learnability from Lower Bounds for Approximate MCSP)**.** *Let $s, t : \mathbb{N} \to \mathbb{N}$ be size functions such that $n \leq s(n) \leq t(n)$ and ε, δ be parameters such that $\varepsilon < 1/2$, $0 \leq \delta < 1/2$. If for infinitely many input lengths $N = 2^n$, $\mathsf{MCSP}[(s, 0), (t, \varepsilon)] \notin \mathsf{SIZE}[N \cdot \mathsf{poly}(s(n), t(n))]$, then for infinitely many inputs n, $\mathsf{SIZE}[s(n)]$ cannot be learnt up to error ε with confidence δ by $t(n)$-size circuits using non-adaptive membership queries over the uniform distribution.*

We also show a related result which gives lower bounds for learnability of P/poly by assuming a lower bound against worst-case MCSP instead.

Lemma 3.7 (Hardness Magnification for Learnability from Lower Bounds for Gap MCSP)**.** *Let $c \geq 1$ be an arbitrary constant. If there is $\varepsilon < 1/2$, such that for infinitely many input lengths $N = 2^n$, $\mathsf{MCSP}[n^c, 2^n/n^{c-2}] \notin \mathsf{SIZE}[N^{1+\varepsilon}]$, then for every $\gamma \in (0,1)$, for infinitely many inputs n, $\mathsf{SIZE}[n^c]$ cannot be learnt up to error $1/n^c$ with confidence $1/n$ by $\mathsf{SIZE}[2^{O(n^\gamma)}]$-circuits using non-adaptive membership queries over the uniform distribution.*

Proof of Theorem 3.1. The following implications establish the desired equivalences.

$(a) \implies (c)$: For the parameters c, γ, ε given by (a), we apply Lemma 3.6 for $s(n) = n^c$ and $t(n) = 2^{n^\gamma}$, to see that for some $\gamma' > 0$, $\mathsf{SIZE}[n^c]$ cannot be learned by circuits of size $2^{O(n^{\gamma'})}$ via non-adaptive queries up to an error $O(1/n^c)$ and confidence $1/n$.

$(c) \implies (d)$: We show the contrapositive of this implication. Suppose that for every $d \geq 1$, there exists a $\mathsf{SIZE}[\mathsf{poly}(n)]$-natural property that is useful against $\mathsf{SIZE}[n^d]$ for all large enough n. By Theorem 3.3, for every $c \geq 1$ and every $\gamma > 0$, we can learn $\mathsf{SIZE}[n^c]$ by a sequence of oracle $\mathsf{SIZE}[2^{O(n^\gamma)}]$-circuits up to an error of n^{-c} and confidence $1/n$, by choosing $d = ac/\gamma$ for the constant a from Theorem 3.3.

$(d) \implies (a)$, $(d) \implies (b)$: Trivial, using the fact that random functions are hard.

$(c) \implies (e)$: Follows from the contrapositive of Theorem 3.5.

$(e) \implies (c)$: Follows from the non-uniform version of Proposition 29 in [OS17], using essentially the same proof.

$(b) \implies (c)$: For the parameter c given by (b), we apply Lemma 3.7 to see that $\mathsf{SIZE}[n^c]$ cannot be learned by circuits of size $2^{O(n^\gamma)}$ via non-adaptive queries up to an error $O(1/n^c)$ and confidence $1/n$, for any $\gamma \in (0,1)$. □

We now complete the proof of Theorem 3.1 by proving Lemmas 3.6 and 3.7.

Proof of Lemma 3.6. For the promise problem $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ over N inputs, define

$$\Pi_{yes} = \{z \in \{0,1\}^N \mid \exists \text{ circuit of size} \leq s(n) \text{ that computes } f_y\}$$
$$\Pi_{no} = \{z \in \{0,1\}^N \mid \text{no circuit of size} \leq t(n) \text{ can } (1-\varepsilon)\text{-approximate } f_y\}$$

We prove the contrapositive of the statement. For a fixed $0 \leq \varepsilon < 1/2$ and $0 \leq \delta < 1/2$, let $\{D^{(\cdot)_n}\}_{n \geq 1} \in \mathsf{SIZE}[t(n)]$ be the corresponding sequence of (randomised) oracle circuits which (ε, δ)-learns $\mathsf{SIZE}[s(n)]$, where $D^{(\cdot)_n}$ makes non-adaptive queries to some function $f \in \mathsf{SIZE}[s(n)]$ over n inputs. The input to $D^{(\cdot)_n}$ is a string of length $t(n)$ which serves as its source of randomness.

We first transform the learner $D^{(\cdot)_n}$ to a deterministic non-uniform "list" learner $E^{(\cdot)_n}$, which is a learner that has non-adaptive access to f and outputs a list of R hypotheses circuits, such that there exists at least one hypothesis in this list which $(1-\varepsilon)$-approximates f. This algorithm is simple: $E^{(\cdot)_n}(w_1, \ldots, w_R)$ runs $R = s(n)^2$ independent copies of the learner $D^{(\cdot)_n}$ (using fresh randomness w_i each time) to learn f, and outputs R hypotheses $h_1, \ldots, h_R$. The probability that none of these hypotheses approximate f_n on at least $(1-\varepsilon)$-fraction of the inputs is at most δ^R. Considering each function in $\mathsf{SIZE}[s(n)]$ as a fixed "input", based on our choice of R, we use Adelman's trick to non-uniformly fix a "good" random string w^* such that for every $f \in \mathsf{SIZE}[s(n)]$, $E^{(\cdot)_n}(w^*)$ outputs a list of hypothesis such that there exists one of them that approximates f_n on at least $(1-\varepsilon)$-fraction of the inputs.

We now construct a circuit C_N which computes $\mathsf{MCSP}[(s,0),(t,\varepsilon)]$ by reducing it to the list learner. On input $z \in \{0,1\}^N$, C_N runs $E^{f_z}(w^*)$ by answering its oracle queries to f_z using its truth table z. Let $h_1, \ldots, h_R$ be the R hypotheses output by the list learner. If the output string of the learner can be interpreted as a $t(n)$-sized circuit, F_N computes the truth tables $\mathsf{tt}(h_1), \ldots, \mathsf{tt}(h_R) \in \{0,1\}^N$. It then compares z with each of these truth tables and outputs 1 if and only if there exists an $i \in [R]$, such that h_i can be interpreted as a $t(n)$-sized circuit and the Hamming distance between $\mathsf{tt}(h_i)$ and z is at most εN.

The correctness of C_N follows from the properties of the list learner. If $z \in \Pi_{yes}$, then f_z is computed by some circuit of size at most $s(n)$. The choice of w^* ensures that for any f_z, there exists an $i \in [R]$, such that the Hamming distance between $\mathsf{tt}(h_i)$ and z is at most εN. On the other hand, if $z \in \Pi_{no}$, then no circuit of size at most $t(n)$ can even $(1-\varepsilon)$-approximate f_z. In particular, each of the h_is (if encoded properly) is a $t(n)$-sized circuit h and is at least ε-far from f_z, and thus F_N outputs 0.

Observe that for each run of D^{f_z}, it is enough to use appropriate literals from the input z whenever it needs to access the truth table of f_z. Indeed, when w_i^* gets fixed non-uniformly, the set of queries made by $D^{(\cdot)_n}$ in the i^{th} run also gets fixed because of its non-adaptivity. Recall that the size of the learner is $t(n)$ and the hypothesis h output by the learner can be interpreted as a $t(n)$-sized circuit and efficiently computed by another circuit of size $\mathsf{poly}(t(n))$. C_N runs $E^{(\cdot)_n}$ of size $O(t(n) \cdot s(n)^2)$ to generate the hypotheses, then generates their truth tables using $N \cdot \mathsf{poly}(t(n)) \cdot s(n)^2$ gates, and uses $O(N \cdot s(n)^2)$ many gates to check if there exists a hypothesis truth table which approximates z well. Thus, C_N has size $O(N \cdot \mathsf{poly}(s(n), t(n)))$ overall. □

Proof sketch of Lemma 3.7. We compute $\mathsf{MCSP}[n^c, 2^n/n^{c-2}]$ by constructing a list learner $\{E^{(\cdot)_n}\}_{n \in N}$ using the learner $\{D^{(\cdot)_n}\}_{n \geq 1}$ and reducing gap MCSP to it. The proof is almost the same as that of Lemma 3.6. The only difference is the acceptance criterion where the

reduction accepts input $z \in \{0,1\}^N$ if and only if there exists an $i \in [R]$, such that the Hamming distance between $\mathsf{tt}(h_i)$ and z is at most $\frac{2^n}{n^c}$.

When the input z is a yes instance, there exists an h_i amongst the list of R hypotheses obtained from E^{f_z} which has error at most $1/n^c$, and C_N accepts it. When the input z is a no instance, any hypothesis h which $D^{(\cdot)_n}$ outputs must have error greater than $1/n^c$. Indeed, if the error is less than $1/n^c$, then by hardwiring all the error inputs by using circuits of size at most $O\left(\frac{2^n}{n^c} \cdot n\right)$ we get a circuit of size at most $2^n/n^{c-2}$, which is a contradiction to the promise of the no instance. In this case, F_N rejects z as none of the R hypotheses have a Hamming distance of at most $\frac{2^n}{n^c}$ with z. Thus, C_N computes $\mathsf{MCSP}[n^c, 2^n/n^{c-2}]$ and has size $O(N \cdot \mathsf{poly}(2^{O(n^\gamma)}, n^c) \leq O(N^{1+\varepsilon})$, for every $\varepsilon > 0$. □

3.4 Towards a More Robust Theory of Magnification

Does magnification for $\mathsf{MCSP}[n^c, 2^{n^\varepsilon}]$ refute natural proofs against $\mathsf{P/poly}$? In this section we observe that this question is connected to basing hardness of learning on worst-case hardness assumptions for NP.

We first show that hardness of learning $\mathsf{P/poly}$ based on worst-case assumptions implies that super-linear lower bounds for worst-case MCSP refutes natural properties against $\mathsf{P/poly}$. Formally,

Proposition 3.8. *Suppose that there exists $\lambda \in (0,1)$ and $k \geq 1$, such that $\mathsf{NP} \not\subseteq \mathsf{SIZE}[2^{O(n^\lambda)}]$ implies that $\mathsf{SIZE}[n^k]$ is hard to learn by circuits of size 2^{n^λ} up to error $1/n^k$. Then, there exists $\varepsilon > 0$ and $c \geq 1$ such that $\mathsf{MCSP}[n^c, n^{c+1}] \notin \mathsf{SIZE}[N^{1+\varepsilon}]$ implies that there exists no $\mathsf{P/poly}$-natural properties useful against $\mathsf{P/poly}$.*

Proof. Suppose that there exists a $\mathsf{P/poly}$-natural property useful against $\mathsf{P/poly}$. From Theorem 3.3, we see that for every $k \geq 1$ and $\lambda \in (0,1)$, there exists a sequence of 2^{n^λ} size circuits which learns $\mathsf{SIZE}[n^k]$ up to error $1/n^k$. From our assumption, this implies that $\mathsf{NP} \subseteq \mathsf{SIZE}[2^{O(n^\lambda)}]$. From Theorem 4 in [OPS19] (or even Theorem 4.38 in Section 4.5), we see that for every $\varepsilon > 0$ and $c \geq 1$, $\mathsf{MCSP}[n^c, n^{c+1}] \in \mathsf{SIZE}[N^{1+\varepsilon}]$. □

Next, we show a reverse implication from the stronger assumption of a reduction from worst-case MCSP to approximate MCSP, instead of assuming the refutation of natural proofs from super-linear lower bounds for worst-case MCSP. Additionally, we also need to assume that MCSP with large parameters is NP-Complete.

Define a polynomial-time reduction A from $\mathsf{MCSP}[s,t]$ to $\mathsf{MCSP}[(s^k, 0), (t^{1/k}, \varepsilon)]$, for any $k \geq 1$, as one which maps instances of $\mathsf{MCSP}[s,t]$ such that: If $f \in \mathsf{SIZE}[s]$, then $A(\mathsf{tt}(f))$ is the truth table of a function in $\mathsf{SIZE}[s^k]$, whereas if $f \notin \mathsf{SIZE}[t]$, then $A(\mathsf{tt}(f))$

is the truth table of a function which cannot even be $(1-\varepsilon)$-approximated by $\mathsf{SIZE}[t^{1/k}]$ circuits.

Proposition 3.9. *Suppose there exists a polynomial-time reduction A from $\mathsf{MCSP}[s,t]$ to $\mathsf{MCSP}[(s^k,0),(t^{1/k},\varepsilon)]$ for some $k \geq 1$, and that for every $0 < \alpha < \beta < 1$, $\mathsf{MCSP}[2^{\alpha n}, 2^{\beta n}]$ is NP-Complete. Then $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ implies that there exists small enough $\alpha > 0$, such that for every β, where $\alpha < \beta/k < 1/k$, $\mathsf{SIZE}[2^{k\alpha n}]$ is hard to learn by $\mathsf{SIZE}[2^{\beta n}]$-size circuits up to error ε.*

Proof. Suppose that for every small enough $\alpha > 0$, there exists $k\alpha < \beta < 1/k$ and a $\mathsf{SIZE}[2^{\beta n}]$-learner for $\mathsf{SIZE}[2^{k\alpha n}]$ up to error ε. Pick any small enough $\alpha > 0$. From Lemma 3.6 we see that, $\mathsf{MCSP}[(2^{k\alpha n},0),(2^{\beta n},\varepsilon)] \in \mathsf{P/poly}$. Using the reduction A, we see that $\mathsf{MCSP}[2^{\alpha n}, 2^{k\beta n}] \in \mathsf{P/poly}$. Since $\mathsf{MCSP}[2^{\alpha n}, 2^{k\beta n}]$ is NP-Complete, we see that $\mathsf{NP} \subseteq \mathsf{P/poly}$. □

3.5 Upper Bounds for THR-AC^0-MCSP

For the rest of this section, we assume that all Boolean functions are over $\{-1,1\}^n$. See Section 2.5 for definitions on the Fourier expansion of a Boolean function.

Recall that a THR (weighted threshold) gate over n variables is a Boolean function $h : \{-1,1\}^n \to \{-1,1\}$ where $h(x) = \mathrm{sign}(\sum_{i=1}^n w_i x_i - \theta)$, where $w_1, \ldots, w_n, \theta$ are arbitrary real values. Note that the majority gate denoted by MAJ is a THR gate such that $w_1 = \cdots = w_n = 1$ and $\theta = 0$.

The main result of this section is a TC^0 upper bound for approximate THR-AC^0-MCSP.

Theorem 3.10. *There exists a universal constant e such that the following holds. For every constant $c \geq 1$, $d \geq 2$, $0 < \varepsilon < 1/2$ and any $\gamma > 0$, THR-AC^0_d-$\mathsf{MCSP}[(n^c,0),(2^{n^\gamma},\varepsilon)] \in \mathsf{TC}^0_{e\cdot d}[N^{1+\rho}]$, for every $\rho > 0$.*

Theorem 3.10 is proved via a generalisation of Lemma 3.6 which (informally) states that barely super-linear lower bounds for approximate $\mathcal{C}$-MCSP against OR-THR-$\mathcal{C}$ implies the non-existence of *self-learners* for $\mathcal{C}$. For constant depth d, $\mathcal{C}_d[s(n)]$ has a self-learner if it can be learnt by $\mathcal{C}_d[t(n)]$-circuits, where $2^n/n^{\omega(1)} \geq t(n) > s(n)$, such that the learner outputs a string which is an *effective encoding* of a $\mathcal{C}_d[t(n)]$-circuit. This means, it should be possible for $\mathcal{C}_{d'}$-circuits to interpret the encoding as that of a $\mathcal{C}_d$-circuit, and efficiently evaluate computations given that description (typically $d' = O(d)$ and $t(n)$ is taken to be much larger than $s(n)$). This is an informal working definition, as for instance we do not consider the dependence on ε or δ in the size of the learner, and we'll consider these details in the formal result statements (cf. [OS18b] for more details).

Lemma 3.11 (Generalisation of Lemma 3.6)**.** *Let $s, t : \mathbb{N} \to \mathbb{N}$ be size functions such that $n \leq s(n) \leq t(n)$ and ε, δ be parameters such that $0 \leq \delta < 1/2$ and $0 < \varepsilon < 1/2$. For any constant $d \geq 2$, let $\mathcal{C}_d$ be any class of circuits of depth d which is closed under composition.*

There exists a universal constant $e > 0$, such that the following holds. If for infinitely many input lengths $N = 2^n$, $\mathcal{C}_d$-$\mathsf{MCSP}[(s, 0), (t, \varepsilon)] \notin \mathsf{OR}$-$\mathsf{THR}$-$\mathcal{C}_{e \cdot d}[N \cdot \mathsf{poly}(s(n), t(n))]$, then for infinitely many n, $\mathcal{C}_d[s(n)]$ cannot be learnt up to error ε with confidence δ by (randomised) $\mathcal{C}_d[t(n)]$-circuits using non-adaptive membership queries over the uniform distribution.[5]

Proof Sketch. The proof follows by inspection of Lemma 3.6. Once again, we construct a sequence of deterministic list learner circuits $\{E^{(\cdot)_n}\}_{n \geq 1}$ using the randomised learner circuits $\{D^{(\cdot)_n}\}_{n \geq 1}$. Let $z \in \{0,1\}^N$ be the input to $\mathcal{C}_d$-MCSP and $R = s(n)^2$. $E^{(\cdot)_n}$ uses non-uniform advice w^* as a "good" random string to run R independent copies of the learner $D^{(\cdot)_n}$, such that for every $f \in \mathcal{C}_d[s(n)]$, E outputs a list of hypotheses $h_1, \ldots, h_R$, in which at least one of them is a good approximation of f. The circuit C_N computing $\mathcal{C}_d$-MCSP runs E^{f_z} by providing it with answers to the queries made to f_z (which are fixed by w^*), and outputs 1 if and only if there exists a hypothesis truth table whose Hamming distance with z is at most εN. We observe the correctness of the reduction, from the proof earlier.

We finish the proof by analysing the structure of C_N. E^{f_z} is computed by running R $\mathcal{C}_d$-circuits in parallel, and thus its depth is also d. C_N then computes the truth tables of $h_1, \ldots, h_R$. Using the fact that $D^{(\cdot)_n}$ outputs an effective encoding of a $\mathcal{C}_d$-hypothesis, we see that circuits of depth $O(d)$ can evaluate the truth tables of all the R hypotheses. C_N then checks whether the Hamming distance between the truth tables of each h_i and z is at most εN, by using a layer of THR gates (along with constantly many layers of gates to check for inequalities in the two strings). Finally, C_N uses an output OR gate to provide the answer. Thus, using the earlier proof, we see that C_N is of size $N \cdot \mathsf{poly}(s(n), t(n))$ and from the construction, it is an OR-THR-$\mathcal{C}_{O(d)}$-circuit. □

Conversely, Lemma 3.11 provides a way of using $\mathcal{C}$-self-learners to derive upper bounds for approximate $\mathcal{C}$-MCSP. Indeed, we give an upper bound for approximate THR-AC^0-MCSP by showing that the learning algorithm for THR-$\mathsf{AC}^0[\mathsf{poly}(n)]$ by Gopalan and Servedio [GS10], can in fact be simulated by sub-exponential sized THR-AC^0 (randomised) circuits.[6]

[5] For clarity, $t(n)$ is actually $t'(n, 1/\delta, 1/\varepsilon)$ for some $t'(n)$ which is $o(t(n))$.

[6] This observation was stated in [OS18b]. We provide a proof of this statement.

The learning algorithm by [GS10] follows the well-known framework by [LMN93] for learning Boolean functions whose Fourier weight is highly concentrated on monomials with low degree t, using random examples over the uniform distribution (called the "low-degree algorithm"). For this, the learning algorithm obtains a close approximation to the Fourier coefficient of each monomial of size at most t empirically, by using sufficiently many independent random examples. It then uses these estimates to obtain a good approximation of the function we are trying to learn. The algorithm outputs the sign of this approximation as its hypothesis.

Lemma 3.12 (Low-degree algorithm [LMN93] (see Section 3.4 in [O'D14]))**.** *Let $\mathcal{C}$ be a class such that every function in $\mathcal{C}$ has Fourier weight $(1-\varepsilon/2)$-concentrated on monomials of degree t. Then $\mathcal{C}$ can be (ε,δ)-learnt using random examples over the uniform distribution in time $\mathsf{poly}(n^t, 1/\varepsilon, \log(1/\delta))$, for any $\varepsilon,\delta>0$.*

Moreover, the learner outputs a hypothesis h_n, such that on input $x\in\{-1,1\}^n$, $h(x)=sign\left(\sum_{S,|S|\leq t}\widetilde{f}(S)\chi_S(x)\right)$, where each $\widetilde{f}(S)$ is an approximate estimate of the Fourier coefficient on S.

In particular, $O(n^t)$ is the number of monomials of size t.

[GS10] show that for any constant depth d, polynomial-sized THR-AC^0 circuit C and any constant $0<\varepsilon<1/2$, the Fourier weight of C is $(1-4\varepsilon)$-concentrated on monomials of degree $t=\frac{(\log s)^{\Theta(d)}\cdot 2^{\Theta(\log^{2/3} s)}}{\varepsilon^{\Theta(\log^{1/3} s)}}=2^{\Theta((\log^{2/3} n))}$. Combining this with the low-degree algorithm, [GS10] then show that THR-AC^0 circuits of depth d and polynomial size can be learnt up to a constant accuracy ε and confidence $1/n$ in time $O(n^t)=O\left(n^{2^{\Theta((\log n)^{2/3})}}\right)\leq O\left(2^{n^\gamma}\right)$, for every $\gamma>0$.

Before moving on to the proof, we need the following well-known result about effective encodings for AC^0. This lemma can in fact be extended to any standard constant-depth circuit class. We provide a proof sketch for completeness.

Lemma 3.13. *For any $s=s(n)$, any constant d and any n-variate size-s depth-d AC^0 circuit C, there exists an encoding σ_C of size $O(s^2)$, for which a sequence of polynomial sized AC^0-circuits take σ_C and $x\in\{0,1\}^n$ as inputs and output $C(x)$.*

Proof. W.l.o.g assume that C is layered i.e. each layer consists of only AND or OR gates, and the NOT gates are pushed to the bottom.[7] Suppose that the layer above the inputs has only OR gates.

We encode C as an adjacency matrix of a directed graph whose vertices are formed by the n inputs, their complements and the s nodes in C (one can also add extra vertices

[7] Any AC^0 circuit C can be transformed into such a circuit by adding dummy gates to make it layered and using De Morgan's laws to push the NOT gates to the bottom. The new size is at most $O(s\cdot d)$ and depth is at most $O(d)$.

for advice inputs if necessary), and the edges are given by the wires in C directed from a vertex in a bottom layer to the top. The size of the matrix is $(2n+s)^2$. Furthermore, the vertices are ordered from the input layer (depth 0) to the output gate at depth d, where any vertices in layer i come before layer $i+1$ in the ordering.

For $m = (2n+s)^2+n$, we first show that if C is a CNF over n inputs, then there exists an AC^0-circuit D_m that takes the description of C and input $x \in \{0,1\}^n$, and outputs $C(x)$. An analogous argument can be used for a DNF. Let u_0 be a Boolean vector of length $2n+s$ such that the first $2n$ bits are fed by x and $\overline{x}$, with the rest set to 0. Let g_1 be an OR gate in the first layer of C and v_{g_1} be the vector obtained from the corresponding matrix column in the encoding. D_m computes the coordinate-wise AND of u_0 and v_{g_1}, and outputs the OR of all these coordinates. Clearly this circuit simulates g_1.

D_m simulates every OR gate in layer 1 this manner and constructs the Boolean vector u_1, such that the outputs from x, $\overline{x}$ and the outputs of the simulations for the OR gates feed into corresponding entries in u_1. D_m next simulates the AND gate g_2 in a similar fashion - by taking the coordinate-wise AND of u_1 and v_{g_2}. However, for every coordinate $j \in [2n+s]$ of v_{g_1} that has 1 in it, the output of the coordinate-wise AND is fed and if not, its complement is fed into an AND gate. D outputs the result of this simulation of g_2 as $C(x)$.

D_m repeats this step inductively layerwise, by considering the outputs of the simulations of the gates from layer $i-1$ as inputs to a gate in layer i, whose connectivity is provided in the matrix. Finishing this simulation for C's output gate in the d^{th} layer, D_m outputs $C(x)$. From our construction, we observe that D_m is an AC^0-circuit of size $\mathsf{poly}(s,n)$ and its depth is $O(d)$. □

Proof of Theorem 3.10. For any $d > 1$, let $\mathcal{C}_d = \mathsf{THR}\text{-}\mathsf{AC}^0{}_d[\mathsf{poly}(n)]$ and $\varepsilon > 0$ be any constant. Let $t = 2^{\Theta(\log^{2/3} n)}$ be the degree of the monomials on which the Fourier weight of any function in $\mathcal{C}_d$ is $(1-\varepsilon/4)$-concentrated (as given by [GS10]) and M be the number of such monomials, where $M = O(n^t) = O\left(n^{2^{\Theta(\log^{2/3} n)}}\right) \leq O\left(2^{n^\gamma}\right)$, for every $\gamma > 0$.

We first construct a sequence of constant depth $\mathcal{C}_d$-circuits $\{D^{(\cdot)_n}\}$ such that for every large enough input length n, $D^{(\cdot)_n}$ learns n-variate $\mathcal{C}_d$-circuits up to constant error ε and confidence $\delta = 1/n$, using random examples over the uniform distribution. The learner $D^{(\cdot)_n}$ does the following:

- For each monomial $\chi_i(x) = \prod_{k \in S_i} x_k$, where $1 \leq i \leq M$ and $S_i \subset [n]$ of size at most t, $D^{(\cdot)_n}$ takes $r = O\left(\frac{M \cdot \log(M/\delta)}{\varepsilon}\right) = O(Mt \log n)$ many random examples and their labels.

- For each random example z_{ij}, where $1 \leq i \leq M, 1 \leq j \leq r$, $D^{(\cdot)_n}$ computes $y_{ij} = f(z_{ij}) \cdot \chi_i(z_{ij}) \in \{-1, 1\}$.

- $D^{(\cdot)_n}$ outputs the description of a hypothesis h_n, into which it hardcodes the values $y_{11}, \ldots, y_{1r}, \ldots, y_{M1}, \ldots, y_{Mr}$ as non-uniform advice. On input $x \in \{-1,1\}^n$, h_n does the following :
  - h_n first computes the parities (or monomials) $\chi_1(x), \ldots, \chi_M(x)$ for all M monomials of size at most t.
  - It then uses a THR gate over $M \cdot r$ many inputs with $w_1 = \cdots = w_{M \cdot r} = 1/r$ and $\theta = 0$, to output -

$$h(x) = \mathsf{THR}(y_{11}\chi_1(x), \ldots, y_{1r}\chi_1(x), \ldots, y_{M1}\chi_M(x), \ldots, y_{Mr}\chi_M(x))$$

To see the correctness of h, we have

$$\begin{aligned} h(x) &= \text{sign}\left(\frac{y_{11}\chi_1(x)}{r} + \cdots + \frac{y_{1r}\chi_1(x)}{r} + \cdots + \frac{y_{M1}\chi_M(x)}{r} + \cdots + \frac{y_{Mr}\chi_M(x)}{r}\right) \\ &= \text{sign}\left(\left(\frac{y_{11}}{r} + \cdots + \frac{y_{1r}}{r}\right) \cdot \chi_1(x) + \cdots + \left(\frac{y_{M1}}{r} + \cdots + \frac{y_{Mr}}{r}\right) \cdot \chi_M(x)\right) \\ &= \text{sign}\left(\widetilde{f}(S_1)\chi_1(x) + \cdots + \widetilde{f}(S_M)\chi_M(x)\right) \end{aligned}$$

Here $\widetilde{f}(S_i) = \left(\sum_{j=1}^{r} y_{ij}\right)/r$ is the empirical estimate of the Fourier coefficient of each monomial of degree at most t (recall from Section 2.5 that any Fourier coefficient $\widehat{f}(S_i) = \mathbf{E}_x[f(x) \cdot \chi_i(x)]$). Thus, h_n approximates C with error at most $\varepsilon/2$ from Lemma 3.12.[8]

It is important to note that h_n *does not* perform division by r to compute the estimates for the Fourier coefficients, but instead this estimation is delegated to the THR gate by setting its weights to $1/r$ each.

Next, we analyse the circuit complexities of $D^{(\cdot)_n}$ and h_n. $D^{(\cdot)_n}$ computes $O(M \cdot r)$ many parities $y_{11}, \ldots, y_{Mr}$, each of which is over at most t many inputs. Thus, $D^{(\cdot)_n}$ is a circuit of size at most $2^{t^{1/d-1}} \cdot M \cdot r \leq 2^{n^\gamma}$, for every $\gamma > 0$.[9] The effective encoding for the circuit based on Lemma 3.13 can be provided as non-uniform advice (for the single threshold gate, the encoding will also have the rational weights $1/r$ in it), and $D^{(\cdot)_n}$ uses this along with $y_{11}, \ldots, y_{Mr}$ to output the $\mathcal{C}$-circuit description of h_n. In other words, $D^{(\cdot)_n}$ can be computed by a $\mathcal{C}$-circuit of 2^{n^γ} size and depth d.

The output is an encoding of a hypothesis h_n, which computes M many parities $\chi_1(x), \ldots, \chi_M(x)$ on at most t inputs, and $O(M \cdot r)$ many constant input parities before feeding them into a THR gate. The circuit size of h_n is at most $M \cdot 2^{t^{1/d-1}} + O(M \cdot r)$ which is also at most 2^{n^γ}, for every $\gamma > 0$, and its depth is d.

[8] In fact we choose r, such that with probability at least $1 - \delta$, all Fourier estimates $\widetilde{f}(S_i)$ are within $\sqrt{\varepsilon/16M}$ of the real Fourier coefficients.

[9] We know that for every $d > 1$, parity over n inputs requires AC^0_d-circuits of size $2^{n^{1/d-1}}$.

Thus, $\mathcal{C}_d$-circuits of polynomial size can be learnt up to error $\varepsilon/2$ and confidence $1/n$ using random examples over the uniform distribution by $\mathcal{C}_d$-circuits of size $2^{n^{o(1)}}$ (which output $\mathcal{C}_d$-circuit hypotheses), which is at most 2^{n^γ} for all $\gamma > 0$. From Lemma 3.11, we see that for every constant c, $\gamma > 0$ and $\rho > 0$, $\mathcal{C}_d$-$\mathsf{MCSP}[(n^c, 0), (2^{n^\gamma}, \varepsilon)]$ is in $\mathsf{TC}^0[N^{1+\rho}]$. □

Remark 3.14. *Assuming the Gotsman-Linial conjecture [GL94] on the total influence of every degree d PTF, [GS10] show a stronger Fourier concentration for* THR-AC^0 *circuits. They get a quasi-polynomial time learner under this assumption (by Lemma 3.12). Using this with Lemma 3.11, we get a strengthening of the upper bound for* THR-AC^0-MCSP-$[(\mathsf{poly}(n), 0), (n^{\mathsf{poly}(\log n)}, \varepsilon)]$.

Remark 3.15. *[GS10] show that* THR-$\mathsf{AC}^0[\mathsf{poly}(n)]$ *can actually be learnt to accuracy $\varepsilon > 1/2^{O((\log n)^{1/3})}$ in time $O(2^{n^{o(1)}})$, if $d \leq O(\log^{2/3} n)/\log\log n$. Similarly, assuming Gotsman-Linial conjecture, they show that constant depth-d* THR-$\mathsf{AC}^0[\mathsf{poly}(n)]$ *can be learnt up to $1/\mathsf{poly}(\log n)$ accuracy in quasi-polynomial time.*

Unfortunately, this technique does not give non-trivial learners for any $\varepsilon = 1/n^c$, where $c \geq 1$, and thus, cannot be extended to show that $\mathcal{C}$-$\mathsf{MCSP}[n^c, 2^n/n^{c-2}]$ *has a super-linear* TC^0 *circuit upper bound. In particular, the [GS10] learners only work when the error is large enough ($\varepsilon \geq 1/n^{o(1)}$), whereas we need $\varepsilon = 1/n^c$ for the worst-case version of Lemma 3.11 to go through.*

3.6 Open Problems and Future Directions

We end this chapter with some technical questions pertaining to the refutation of natural proofs by hardness magnification. First, we would like to understand if it is possible to strengthen items (a) and (b) in Theorem 3.1 to a wider range of parameters. For example, is hardness magnification for worst-case $\mathsf{MCSP}[n^c, 2^{n^\gamma}]$ non-naturalisable? The core of this question seems to be the problem of reducing worst-case MCSP from Item (a) to approximate MCSP from Item (b) (see Section 3.4).

Another direction is to show that hardness magnification refutes natural proofs also in the context of *non*-meta-computational problems. Interestingly, many magnification theorems from [OPS19] established for MCSP and variants were subsequently shown to hold for any sparse family of languages in NP [CJW19]. Could it be the case that hardness magnification avoids the natural proofs barrier in a much broader sense?

Finally, from the point of view of self-learning, an interesting question is to explore what circuit classes are self-learnable (cf. [OS18b]). As a starting point, it could be worthwhile to explore the existence of quasi-polynomial sized AC^0-learners for AC^0.

Chapter 4

The Locality Barrier for Hardness Magnification

4.1 Introduction

In this chapter, we are interested in understanding the prospects of proving new lower bounds using hardness magnification, including potential barriers.

4.1.1 Hardness Magnification Frontiers

While hardness magnification is a broad phenomenon, its most promising instantiations seem to occur in the setting of circuit classes such as NC^1. The potential of hardness magnification stems from establishing the following scenario.

HM Frontier: There is a natural problem Q and a computational model $\mathcal{C}$ such that:

1. (*Magnification*) $Q \notin \mathcal{C}$ implies $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ or a similar breakthrough.
2. (*Evidence of Hardness*) $Q \notin \mathcal{C}$ under a standard conjecture.
3. (*Lower Bound against* $\mathcal{C}$) $L \notin \mathcal{C}$, where L is a simple function like PARITY.
4. (*Lower Bound for* Q) $Q \notin \mathcal{C}^-$, where $\mathcal{C}^-$ is slightly weaker than $\mathcal{C}$.

A frontier of this form provides hope that the required lower bound in Item 1 is true (thanks to Item 2), and that it might be within the reach of known techniques (thanks to Items 3 and 4, which provide evidence that we can analyse the circuit model and the problem). HM frontiers have been already achieved in earlier works with a striking example appearing in [OPS19] (see also [CJW19]). Despite the number of works in this area, we note that a HM frontier is achieved only by some magnification theorems (Item 3 is often unknown; e.g. in the case of results in [AK10, CT19]).

In order to make our subsequent discussion more concrete, we provide five examples of HM frontiers. Some of these results are new or require an extension of previous work, and the relevant statements will be explained in more detail in their corresponding sections. The list of frontiers is not meant to be exhaustive, but we have tried to cover different computational models.

(A) HM Frontier for $\mathsf{MKtP}[n^c, 2n^c]$ and AC^0-XOR:

A1. If $\mathsf{MKtP}[n^c, 2n^c] \notin \mathsf{AC}^0\text{-}\mathsf{XOR}[N^{1.01}]$ for large enough $c > 1$, then $\mathsf{EXP} \not\subseteq \mathsf{NC}^1$ (Section 4.3.1).

A2. $\mathsf{MKtP}[n^c, 2n^c] \notin \mathsf{P/poly}$ for large enough c, under exponentially secure PRFs [RR97].

A3. $\mathsf{Majority} \notin \mathsf{AC}^0\text{-}\mathsf{XOR}[2^{N^{o(1)}}]$ (immediate from [Raz87, Smo87]).

A4. $\mathsf{MKtP}[n^c, 2n^c] \notin \mathsf{AC}^0$, for any sufficiently large constant c (Section 4.3.1).

A. $\mathsf{MKtP}[s, t]$ refers to the gap version of MKtP on inputs of length $N = 2^n$ (see Section 2.4). The AC^0-XOR model is an extension of AC^0 where gates at the bottom layer of the circuit can compute arbitrary parity functions. $\mathsf{AC}^0\text{-}\mathsf{XOR}[s]$ denotes AC^0-XOR circuits of size s where the size is measured as the number of gates. This circuit class has received some attention in recent years (cf. [CGJ+18]), and a few basic questions about AC^0 circuits with parity gates (such as constructing PRGs of seed length $o(n)$ and learnability using random examples) remain open for AC^0-XOR as well.

(B) HM Frontier for $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ and $\mathsf{Formula}$-XOR:

B1. $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula}\text{-}\mathsf{XOR}[N^{1.01}]$ implies $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$ (Section 4.4.1).

B2. $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{P/poly}$ under standard cryptographic assumptions [RR97].

B3(a). $\mathsf{InnerProduct} \notin \mathsf{Formula}\text{-}\mathsf{XOR}[N^{1.99}]$ (immediate consequence of [Tal17a]).

B3(b). $\mathsf{MCSP}[2^n/n^4] \notin \mathsf{Formula}\text{-}\mathsf{XOR}[N^{1.99}]$ ([KKL+20]).

B4. $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula}[N^{1.99}]$ ([HS17]; see also [OPS19]).

B. In the statements above, NQP stands for nondeterministic quasi-polynomial time, $\mathsf{InnerProduct}$ is the Boolean function defined as $\mathsf{InnerProduct}(x, y) = \sum_i x_i \cdot y_i \pmod 2$, where $x, y \in \{0,1\}^N$, $\mathsf{Formula}\text{-}\mathsf{XOR}[s]$ refers to the class of Boolean formulas over the De Morgan basis with at most s leaves, where each leaf is an XOR of arbitrary arity over the inputs,[1] and $\mathsf{MCSP}[s, t]$ is worst-case gap MCSP over inputs of size N (see Section 2.3).

[1] Note that $\mathsf{Formula}\text{-}\mathsf{XOR}[N^{1.01}] \subseteq \mathsf{Formula}[N^{3.01}]$. A better understanding of the former class is therefore necessary before we can understand the power and limitations of super-cubic formulas, which is a major open question in circuit complexity.

Intriguingly, as a consequence of A1 and A4 (or B1 and B4), we seem to only be a layer of XOR gates away from proving major breakthroughs in complexity theory!

(C) HM Frontier for $\mathsf{MCSP}[2^{n^{1/2}}/10n, 2^{n^{1/2}}]$ and Almost-Formulas:

C1. $\mathsf{MCSP}[\frac{2^{n^{1/2}}}{10n}, 2^{n^{1/2}}] \notin N^{0.01}$-$\mathsf{Almost\text{-}Formula}[N^{1.01}]$ implies $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ (Section 4.5.1).

C2. $\mathsf{MCSP}[\frac{2^{n^{1/2}}}{10n}, 2^{n^{1/2}}] \notin \mathsf{P/poly}$ under standard cryptographic assumptions [RR97].

C3. $\mathsf{Parity} \notin N^{0.01}$-$\mathsf{Almost\text{-}Formula}[N^{1.01}]$ (Section 4.5.1).

C4. $\mathsf{MCSP}[2^{n^{1/2}}/10n, 2^{n^{1/2}}] \notin \mathsf{Formula}[N^{1.99}]$ ([HS17]; see also [OPS19]).

C. An almost-formula is a circuit with a bounded number of gates of fan-out larger than 1. More precisely, a γ-$\mathsf{Almost\text{-}Formula}[s]$ is a circuit containing at most s many AND, OR, NOT gates of fan-in at most 2, and among such gates, at most γ of them have fan-out larger than 1. Consequently, this class naturally interpolates between formulas and circuits. This magnification frontier can be seen as progress towards establishing magnification theorems for worst-case variants of MCSP in the regime of sub-quadratic formulas (see the discussion in [OPS19]).

(D) HM Frontier for $\mathsf{MCSP}[2^{\sqrt{n}}]$ and one-sided error randomised formulas:

D1. $\mathsf{MCSP}[2^{\sqrt{n}}] \notin \mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{2.01}] \Rightarrow \mathsf{NQP} \notin \mathsf{NC}^1$ (Section 4.6.1).

D2. $\mathsf{MCSP}[2^{\sqrt{n}}] \notin \mathsf{P/poly}$ under standard cryptographic assumptions [RR97].

D3(a). $\mathsf{Andreev}_N \notin \mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{2.99}]$ (implicit in [Hås98]).

D3(b). $\mathsf{MCSP}[2^n/n^4] \notin \mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{2.99}]$ (implicit in [CKLM19]).

D4. $\mathsf{MCSP}[2^{\sqrt{n}}] \notin \mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{1.99}]$ [CJW20].

D. GapAND_N is the promise function on N bits such that it outputs 1 when all input bits are 1, and outputs 0 when at most 1/10 of the input bits are 1. $\mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[s]$ denotes circuits with $\mathsf{GapAND}_{O(N)}$ gate at the top with its inputs being formulas of size s. Therefore, $\mathsf{GapAND}_{O(N)}$-Formula can be seen as *randomised formulas with one-sided error*.[2] The most interesting aspect of this magnification frontier is that the gap between the known hardness result and the magnification threshold is *arbitrarily-close* i.e. we

[2]Suppose there is a $\mathsf{GapAND}_{O(N)}$-Formula circuit computing a function $f : \{0,1\}^N \to \{0,1\}$. Consider a uniform distribution of all sub-formulas below the top $\mathsf{GapAND}_{O(N)}$ gate. Then for any input x, if $f(x) = 1$ then a sample formula from that distribution always outputs 1 on x, otherwise it outputs 0 with probability at least 0.9, if $f(x) = 0$. On the other hand, it is possible to derandomise a distribution of formulas computing f with one-sided error using a top $\mathsf{GapAND}_{O(N)}$ gate. See Section 4.6 for more details.

have an $N^{2-\varepsilon}$ unconditional lower bound for $\mathsf{MCSP}[2^{\sqrt{n}}]$, whereas only an $N^{2+\varepsilon}$ lower bound is needed for the same problem to achieve magnification.[3]

(E) HM Frontier for $(n-k)$-Clique and AC^0:

E1. If $(n-k)$-$\mathsf{Clique} \notin \mathsf{AC}^0[m^{1.01}]$ for $k = (\log n)^C$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ (Section 4.7.1).
E2. (Non-uniform) ETH $\Rightarrow$ $(n-k)$-$\mathsf{Clique} \notin \mathsf{P/poly}$ for $k = (\log n)^C$ (Section 4.7.1).
E3. $\mathsf{Parity} \notin \mathsf{AC}^0$ [Ajt83, FSS84, Yao85, Hås86].
E4. $(n-k)$-$\mathsf{Clique} \notin \mathsf{mP/poly}$ for $k = (\log n)^C$ ([AJ08]; see Section 4.7.1).

E. The ℓ-Clique problem is defined on graphs on n vertices in the adjacency matrix representation of size $m = \Theta(n^2)$. (The statements above refer to the regime of very large clique detection.) The class $\mathsf{mP/poly}$ refers to monotone circuits of polynomial size. In this frontier we modify Item 4 from the notion of an HM frontier, so that instead of slightly weaker $\mathcal{C}^-$ we consider an incomparable $\mathcal{C}^-$.

This frontier is however particularly interesting, as items E1 and E4 connect hardness magnification to a basic question about the power of *non-monotone* circuits when computing *monotone* functions (see [COS17, GKRS19] and the references therein): Is every monotone function in AC^0 computable by a monotone (unbounded depth) Boolean circuit of polynomial size? If this is the case, $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ would follow.

Note that these hardness magnification frontiers offer different approaches to proving lower bounds against NC^1.

Do magnification theorems bring us closer to strong circuit lower bounds? To further understand the limits and prospects of hardness magnification, the following questions are relevant.

Q1. *Extending known lower bounds.* Can we adapt an existing lower bound proof from Items 3 and 4 in some HM frontier to show the lower bound required in Item 1 in that HM frontier? Is it possible to establish the required lower bounds via a reduction from L to Q?

Q2. *Improving existing magnification theorems.* Can we close the gap between Items 1 and 4 in some HM frontiers by establishing a magnification theorem that meets *known* lower bounds, such as the ones appearing in Item 4?

[3] This tight threshold was first observed in [CJW20] via a very similar frontier for $\mathsf{MCSP}[2^{\sqrt{n}}]$ and two-sided error randomised formulas. We include it here to study its potential for proving strong lower bounds.

4.1.2 The Locality Barrier

The results from Chapter 3 show that hardness magnification can go beyond natural proofs. Is there another barrier that makes it difficult to establish lower bounds via magnification? In this section, we present a general argument to explain why the lower bound techniques behind A3-E3, A4-D4 in the magnification frontiers from Section 4.1.1 cannot be adapted (without significantly new ideas) to establish the required lower bounds in Items A1-E1, respectively. We refer to it as the *locality barrier*. While we will focus on these particular examples to make the discussion concrete, we believe that this barrier applies more broadly.

In order to explain the locality barrier, let's consider the argument behind the proof of B1 presented in Section 4.6.1 (or an alternative proof in Section 4.4.1). Recall that this magnification result shows that if $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula\text{-}XOR}[N^{1.01}]$ then $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$. This and other known hardness magnification theorems are established in the contrapositive. The core of the argument is to prove that there are highly efficient $\mathsf{Formula\text{-}XOR}$ circuits that reduce an input to $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ of length N to deciding whether certain strings of length N' (much smaller than N) belong to a certain language L'. Then, under the assumption that $\mathsf{NQP} \subseteq \mathsf{NC}^1$, one argues that L' has polynomial size formulas. Finally, since $N' \ll N$, we can employ such formulas and still conclude that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ is in $\mathsf{Formula\text{-}XOR}[N^{1.01}]$, which completes the proof.

Note that the argument above provides a *conditional* construction of highly efficient formulas for the original problem. Crucially, however, we can derive an *unconditional* circuit upper bound from this argument: If we stop right before we replace the calls to L' by an algorithm for L' (this is what makes the reduction conditional), it unconditionally follows that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ can be computed by highly efficient $\mathsf{Formula\text{-}XOR}$ circuits containing oracle gates of small fan-in, for some oracle. Similarly, one can argue that the problems in Items A1-E1 can be computed in the respective models by highly efficient Boolean devices containing oracles of small fan-in.

We stress that, as opposed to a magnification theorem, where one cares about the complexity of the oracle gates, in our discussion of the locality barrier we only need the fact that there is *some way* of setting these oracle gates so that the resulting circuit or formula solves the original problem. (A definition of this model appears in Section 4.2.4). A more exhaustive interpretation of magnification theorems as constructions of circuits with small fan-in oracles can be found in Section 4.2.5.

On the other hand, we argue that the lower bound arguments from Items A3-E3 of the hardness magnification frontiers quite easily handle (in the respective models) the presence of oracles of small fan-in, *regardless of the function* computed by these oracles. Using a more involved argument we can also localise lower bounds from items A4-D4.

Consequently, these methods do not seem to be refined enough to prove the lower bounds required by A1-D1 without excluding oracle circuits that are unconditionally known to exist for the corresponding problems.

Following the example above, we state our results for the Magnification Frontier B.

Theorem 4.1 (Locality Barrier for HM Frontier B)**.** *The following results hold.*

(B1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) *There exists oracle $\mathcal{O}$, such that for any $\varepsilon > 0$,* $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \in \mathsf{Formula}\text{-}\mathcal{O}\text{-}\mathsf{XOR}[N^{1.01}]$*, where every oracle gate has fan-in at most N^{ε} and appears in the layer right above the* XOR *leaves.*

(B3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques) *For any $\delta > 0$ and oracle $\mathcal{O}$,* InnerProduct *over N input bits cannot be computed by $N^{2-2\delta}$-sized* $\mathsf{Formula}\text{-}\mathcal{O}\text{-}\mathsf{XOR}$ *circuits with at most $N^{2-3\delta}$ oracle gates of fan-in N^{δ} in the layer right above the* XOR *leaves.*

(B4$^{\mathcal{O}}$) (Extension of Lower Bound Techniques Below Magnification Threshold) *There exists a universal constant c such that for all constants $\varepsilon > 0$ and every oracle $\mathcal{O}$,* $\mathsf{MCSP}[n^c, 2^{\varepsilon n/3}] \notin \mathsf{Formula}_3^{\mathcal{O}}[N^{2-\varepsilon}]$*, where the adaptivity of the oracle formulas is at most $o(\log N / \log\log N)$.*

Here, $\mathsf{Formula}_t^{\mathcal{O}}[s]$ denotes the class of functions which can be computed by a sequence of oracle formulas, such that on replacing each oracle gate $\mathcal{O}$ with fan-in B by an equivalent formula of size B^t which reads each of its B inputs exactly B^{t-1} times, we obtain a De Morgan formula of size at most s (see Section 4.8.2 for the motivation of this definition). Further, adaptivity of an oracle formula refers to the maximum number of oracle gates encountered in any path from the root to a leaf.

The first two items of Theorem 4.1 are proved in Section 4.4.2. The third item is proved in Section 4.8.2. While Theorem 4.1 does not specify that, we actually localise all proofs of the lower bounds in B3 and B4 we are aware of. Interestingly, the localisation of B4 allows us to refute the Anti-Checker Hypothesis from [OPS19] (and a family of potential hardness magnification theorems), in Section 4.8.2. We refer to Sections 4.3.2, 4.5.2, 4.6.2, 4.7.2 and 4.8 for analogous statements describing the locality barrier for frontiers A, C, D, and E, respectively.

The recent HM Frontier introduced by [CJW20] for $\mathsf{MCSP}[2^{\sqrt{n}}]$ and two-sided error randomised formulas is also subject to the locality barrier based on ideas analogous to the barrier for HM Frontier D (cf. Section 4.6.2). In another recent work, [CHMY20] (and [MMW19]) prove magnification for $\mathsf{MCSP}[2^{o(n)}]$ against one-tape Turing machines by constructing highly efficient *uniform* oracle algorithms that make short oracle queries.

They also prove a sub-quadratic time lower bound for $\mathsf{MCSP}[2^{(1-o(1))n}]$ against one-tape Turing machines and show that this technique can be localised; it is impossible to extend this technique to $\mathsf{MCSP}[2^{o(n)}]$, with the intention of obtaining magnification (or obtaining magnification from $\mathsf{MCSP}[2^{(1-o(1))n}]$ lower bounds against one-tape Turing machines using similar ideas).

Locality of Computations and Lower Bound Techniques. The fact that many lower bound techniques extend to computational devices with oracles of small fan-in was already observed by Yao in 1989, in a paper on local computations [Yao89]. According to Yao, a local function is one that can be efficiently computed using only localised processing elements. In our terminology, this corresponds to circuits with oracles of small fan-in. Among other results, [Yao89] argues that Razborov's monotone circuit size lower bound for k-Clique [Raz85] and Karchmer and Wigderson's monotone formula size lower bound for $\mathsf{ST\text{-}CONN}$ [KW90] extend to Boolean devices with monotone oracles of bounded fan-in. Compared to Yao's work, our motivation and perspective are different. While Yao is particularly interested in lower bounds that can be extended in this sense (see e.g. Sections 2 and 6 in [Yao89]), here we view such extensions as a *limitation* of the corresponding arguments, meaning that they are not refined enough to address the locality barrier.[4]

We note, however, that not every lower bound technique extends to circuits with small fan-in oracles.[5] For instance, by the work of Allender and Koucký [AK10] (also a more recent work by Chen and Tell [CT19]), the parity function Parity_n over n input bits can be computed by a TC^0 circuit of size $O(n)$ (number of wires) containing $\leq n^{1-\varepsilon}$ oracle gates of fan-in $\leq n^{\varepsilon}$, provided that its depth $d = O(1/\varepsilon)$. On the other hand, it is known that $\mathsf{Parity}_n \notin \mathsf{TC}^0_d[n^{1+c^{-d}}]$ for a constant $c > 0$ [IPS97] (again, the complexity measure is the number of wires). Since the latter lower bound is super-linear for every choice of d, it follows by the result of [AK10, CT19] that it cannot be extended to circuits containing a certain number of oracles of fan-in n^{ε}, for a large enough depth d that depends on ε. Incidentally, the hardness magnification theorems of [AK10, CT19] do not achieve a magnification frontier.

In Section 4.6.2 we identify one specific lower bound related to HM frontier D which is both above the magnification threshold and provably non-localisable (cf. Theorem 4.48). In principle, there might be ways to overcome the locality barrier and match the lower bound with the magnification threshold. We refer to Section 4.9 for additional discussion.

[4] On a more technical level, we are interested in the regime of barely super-linear size circuits and formulas, and our results do not impose a monotonicity constraint on the oracle.

[5] Of course any such discussion depends on parameters such as number of oracles and their fan-ins, so whether a technique avoids the locality barrier or not is relative to a particular magnification theorem.

On Lower Bounds Through Reductions. The discussion above has focused on the possibility of *directly* adapting existing lower bounds from Item 3 in HM frontier to establish the desired lower bound in Item 1. There is however an *indirect* approach that one might hope to use: *reductions*. For instance, in the context of the HM Frontier B discussed above, can we have a reduction from InnerProduct to $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ that would allow us to show that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula\text{-}XOR}[N^{1.01}]$? The first thing to notice is that, for this approach to make sense, the reduction needs to have a specific form and satisfy certain complexity constraints, so that composing the reduction with a small-sized candidate Formula-XOR circuit for $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ violates the hardness of InnerProduct. Is there any hope to design a reduction of this form?

The locality barrier presents a *definitive answer* in this case. Indeed, it is immediate from the first two items of Theorem 4.1 that such a reduction *does not exist* (see Section 4.4.3). For the same reason, it is not possible to use reductions to establish the required lower bounds in some other magnification frontiers. Essentially, under the constraints needed for a reduction to be meaningful, we end up with a class of reductions which produce circuits that are ruled out by the locality barrier.

4.2 Preliminaries

We refer the reader to Chapter 2 for notational conventions and additional definitions.

4.2.1 Pseudorandomness

Definition 4.2 (Pseudorandom Generators)**.** *Let $G : \{0,1\}^r \to \{0,1\}^n$ be a function, $\mathcal{C}$ be a class of Boolean functions and $0 < \varepsilon < 1$. We say that a pseudorandom generator G of seed length r, ε-fools $\mathcal{C}$ if for every function $f \in \mathcal{C}$, we have*

$$\left|\mathbf{E}_{z\sim U_r}[f(G(z))] - \mathbf{E}_{x\sim U_n}[f(x)]\right| \leq \varepsilon$$

Next, we look at particular instantiations of pseudorandom generators (or pseudorandom distributions). We call G as an *ε-biased generator* if it ε-fools parities.

Definition 4.3 (Small-bias generator [NN93])**.** *For any $\varepsilon \in \left(0, \frac{1}{2}\right)$, a generator $G : \{0,1\}^r \to \{0,1\}^n$ is ε-biased if for all non-zero $v \in \{0,1\}^n$,*

$$\left|\mathbf{E}_{z\sim U_r}[\langle v, G(z)\rangle \pmod 2] - \mathbf{E}_{x\sim U_n}[\langle v, x\rangle \pmod 2]\right| \leq \varepsilon$$

In other words, $\mathbf{E}_{z\sim U_r}[\langle v, G(z)\rangle] \in (1/2 - \varepsilon, 1/2 + \varepsilon)$. We refer to the output distribution of G as an ε-biased distribution.

Lemma 4.4 (Construction of ε-biased generators [AGHP92, NN93])**.** *There exists an ε-biased distribution $G : \{0,1\}^r \to \{0,1\}^n$ that can be sampled in time $\mathsf{poly}(n/\varepsilon)$ time deterministically, using an $r = O(\log(n/\varepsilon))$-long seed. Moreover, there exists a sequence of circuits of size at most $\widetilde{O}(\log(n/\varepsilon) \cdot \log n)$, such that given the seed z and an index $j \in [n]$, the circuit prints the j^{th}-bit of $G(z)$.*

A multidimensional distribution is called k-wise independent if any k coordinates of the distribution are uniformly distributed.

Definition 4.5 (k-wise independence)**.** *A distribution over $[m]^n$ is called k-wise independent if for any $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n$ and every $b_1, \ldots, b_k \in [m]$, we have*

$$\Pr\{X_{i_1} = b_1, X_{i_2} = b_2, \ldots, X_{i_k} = b_k\} = 1/m^k$$

We say that a function G is a k-wise independent generator if for a random seed, the output distribution of G is k-wise independent.

Finally, note that any k-wise independent distribution is also a pairwise independent distribution. We state the following version of Chebyshev's inequality (Lemma 2.6) to derive tail bounds on the number of 1s in the output of a pairwise independent distribution over $\{0,1\}^N$.

Lemma 4.6 (Chebyshev's inequality for pairwise independent distributions)**.** *Suppose $X_1, \ldots, X_n$ are pairwise independent random variables over $\{0,1\}$. Then,*

$$\Pr\left\{\left|\sum_{i=1}^{n} X_i - \mathbf{E}\left[\sum_{i=1}^{n} X_i\right]\right| > t\right\} \leq \frac{\sum_{i=1}^{n} \mathsf{Var}[X_i]}{t^2}$$

We refer [Vad12] for more details on these notions.

4.2.2 Random restrictions

Let f be an n-variate Boolean function. A restriction ρ is defined as a mapping from the n variables to $\{0,1,*\}$. We refer to $f|_\rho(x)$ as the restricted Boolean function over variables in $\rho^{-1}(*)$, defined as $f|_\rho(x) = f(y)$, where $y_i = x_i$ if $\rho(i) = *$ and otherwise, $y_i = \rho(i)$.

Any distribution $\mathcal{D}$ over restrictions is called a *random restriction*. We say that $\mathcal{D}$ is p-regular if each variable is unrestricted with probability p. Let $\mathcal{R}_p$ be the p-regular *truly* random restriction where each coordinate of the restriction is independently unrestricted with probability p and otherwise, equal to 0 or 1, with probability $(1-p)/2$ each.

We use the following encoding for a restriction suggested by [TX13].

Definition 4.7 (Restriction)**.** *For any $q, n \in \mathbb{N}$, any binary string $w \in \{0,1\}^{(q+1)n}$ defines a restriction $\rho_w : [n] \to \{0,1,*\}$ as follows. Partition w into $(w_1, \alpha_1), \ldots, (w_n, \alpha_n)$, where $w_i \in \{0,1\}^q$ and $\alpha_i \in \{0,1\}$ for all $1 \leq i \leq n$. Set $\rho_w(i) = *$ if $w_i = 1^q$, otherwise set $\rho_w(i) = \alpha_i$.*

Observe that $\mathcal{R}_p$ is now a uniform distribution over $\{0,1\}^{(q+1)n}$, where $p = 2^{-q}$.

A subclass of random restrictions is a *pseudorandom restriction*, which is the output distribution of a suitable pseudorandom generator over $\{0,1\}^{(q+1)n}$ using much fewer random bits than a truly random restriction. For example, [TX13, Tal17b] use a small-bias generator over $\{0,1\}^{(q+1)n}$ to sample a pseudorandom restriction.

Another way to do this is by making a "pseudorandom selection" obtained from a k-wise independent distribution on $[1/p]^n$ (by sampling a string of length qn). In such a case, each variable is unrestricted with probability p, and any k elements in the output are independent. This is followed by sampling a string from a k-wise independent distribution over $\{0,1\}^n$ to set the restricted variables. More formally, we need the following lemma stating that one can sample a k-wise independent random restriction using a short seed, and moreover all restrictions have a small circuit description.

Lemma 4.8 ([IMZ19, Vad12])**.** *There exists a p-regular k-wise independent random restriction $\mathcal{D}$ distributed over $\rho : [N] \to \{0,1,*\}$, samplable using $O(k \log N \log(1/p))$ bits. Furthermore, each output coordinate of the pseudorandom restriction can be computed in time polynomial in the number of random bits.*

We can compose two restrictions ρ_1 and ρ_2 in a natural way, by applying them one after the other. The *composition* is denoted by $\rho_1 \circ \rho_2$.

4.2.3 Lower bounds using random restrictions

In this section we recall the main ideas behind classical lower bound results based on the random restriction technique, used against circuit classes like AC^0 and cubic-sized De Morgan formulas. We refer to [AB09, Juk12] for more details about the lower bound proofs.

AC^0 circuits: A *Decision Tree* is a tree in which each internal node is labeled by a variable and each leaf is labeled by a Boolean value. A decision tree defines the computation of a Boolean function as a traversal down the tree, by examining the variable at each node and picking a branch based on its value. The output of the function is the leaf label at the end of the traversal. Every Boolean function can be computed by a decision tree. For any function f over n variables, let $\mathsf{DT}_{depth}(f)$ be the depth of the smallest decision tree computing f.

A sequence of works [Ajt83, FSS84, Yao85, Hås86] used the method of random restrictions to show lower bounds for parity against AC^0. The main idea behind the lower bound was the switching lemma:

Lemma 4.9 (Håstad's switching lemma [Hås86])**.** *Let F be a CNF of width w (one which has at most w variables in each clause), $1 > p > 0$, $t > 0$ and ρ be a p-regular random restriction sampled from $\mathcal{R}_p$. Then,*

$$\Pr_{\rho \sim \mathcal{R}_p} \{\mathsf{DT}_{depth}(F|_\rho) > t\} \leq (5pw)^t$$

In particular, under the application of a p-regular truly random restriction for suitably large p, any AC^0-circuit can be shown to collapse to a constant function (with high probability) by iterative application of the switching lemma.

De Morgan formulas Random restrictions have also been used to show lower bounds against small-sized De Morgan formulas. The lower bound is proved by the following classical shrinkage result.

Theorem 4.10 (Formula shrinkage under random restrictions [Hås98, Tal14])**.** *Let f be a Boolean function. For any $p > 0$,*

$$\mathbf{E}_{\rho \sim \mathcal{R}_p}[L(f|_\rho)] \leq O\left(p^2 L(f) + p\sqrt{L(f)}\right)$$

In other words, under the application of a p-regular truly random restriction, the formula size of a function shrinks by at least a p^2-factor in expectation.

4.2.4 Local Circuit Classes

Our definition of local computation is somewhat similar to some definitions appearing in [Yao89].

Definition 4.11 (Local circuit classes)**.** *Let $\mathcal{C}$ be a circuit class (such as $\mathsf{AC}^0[s]$, $\mathsf{TC}^0[s]$, $\mathsf{SIZE}[s]$, etc). For functions $q, \ell, a \colon \mathbb{N} \to \mathbb{N}$, we say that a language L is in $[q, \ell, a]$–$\mathcal{C}$ if there exists a sequence $\{E_n\}$ of oracle circuits for which the following holds:*

(i) Each oracle circuit E_n is a $\mathcal{C}$-circuit.

(ii) There are at most $q(n)$ oracle gates in E_n, each of fan-in at most $\ell(n)$, and any path from an input gate to an output gate encounters at most $a(n)$ oracle gates.

(iii) There exists a language $\mathcal{O} \subseteq \{0,1\}^*$ such that the sequence $\{E_n^{\mathcal{O}}\}$ (E_n with its oracle gates set to $\mathcal{O}$) computes L.

In the definition above, q stands for *quantity*, ℓ for *locality*, and a for *adaptivity* of the corresponding oracle gates.

4.2.5 Hardness Magnification Through the Lens of Oracle Circuits

In Section 3.1.1 we had reviewed some existing magnification results. Following our definition of local circuit classes, we can view these results as *unconditional* upper bounds on the size of small fan-in oracle circuits solving the corresponding problems, for a certain choice of oracle gates. In a magnification theorem, it is important to upper bound the uniform complexity of the oracle gates. For our discussion, this is not going to be relevant.

The fact that existing magnification theorems produce such circuits is a consequence of the algorithmic nature of the underlying proofs, which show how to reduce an instance of a problem to shorter instances of another related problem. By inspection of each proof, it is possible to establish a variety of upper bounds. We explicitly state some of them below.

Proposition 4.12. *The following results hold.*

- [AK10] *For every* $\Pi \in \{\mathsf{BFE}, \mathsf{W}_{\mathsf{S}_5}, \text{W5-STCONN}\}$ *and every* $\beta > 0$*, we have that* $\Pi_n \in \left[O\left(n^{1-\beta}\right), n^{\beta}, O\left(\frac{1}{\beta}\right)\right]\text{–}\mathsf{TC}^0[O(n)]$.
- [LW13] *For every* $\delta > 0$, $\mathsf{CircEval}_n \in \left[n \cdot \mathsf{poly}(\log n), n^{\delta}, n^{1-\delta}\right]\text{–}\mathsf{SIZE}[n \cdot \mathsf{poly}(\log n)]$.
- [OS18a] *For every constructive function* $n \leq s(n) \leq 2^n/\mathsf{poly}(n)$ *and parameter* $0 < \delta(n) < 1/2$, $\mathsf{MCSP}[(s,0),(s,\delta)] \in [N, \mathsf{poly}(s/\delta), 1]\text{–}\mathsf{Formula}[N \cdot \mathsf{poly}(s/\delta)]$.
- [OS18a] *Let* $k = (\log n)^C$*, where* $C \in \mathbb{N}$*. Then* k-Vertex-Cover $\in [1, (\log n)^{4C}, 1]\text{–}\mathsf{AC}^0_d[m^{1+\varepsilon_d}]$*, where* $\varepsilon_d \to 0$ *as* $d \to \infty$.
- [OPS19] *For every* $\beta > 0$ *and any* $s(n) \leq 2^{\beta n}$, $\mathsf{MKtP}[s(n), 2s(n)] \in [N, \mathsf{poly}(s), 1]\text{–}\mathsf{Formula\text{-}XOR}[N \cdot \mathsf{poly}(s)]$.
- [OPS19] *For every size function* $s(n) \leq 2^n/\mathsf{poly}(n)$*, we have* $\mathsf{MCSP}[s(n)/n, s(n)] \in [N \cdot \mathsf{poly}(s), \mathsf{poly}(s), \mathsf{poly}(s)]\text{–}\mathsf{SIZE}[N \cdot \mathsf{poly}(s)]$.
- [MMW19] *For every constructive function* $s(n) \leq 2^n/\mathsf{poly}(n)$*, we have* $\mathsf{MCSP}[s(n)] \in [O(N/\mathsf{poly}(s)), \mathsf{poly}(s), O(n/\log(s))]\text{–}\mathsf{SIZE}[N/\mathsf{poly}(s)]$.

We stress however that not every hardness magnification theorem needs to lead to an unconditional construction of efficient oracle circuits. (All the proofs that we know of produce such circuits though.)

4.3 HM Frontier for Gap MKtP and AC^0-XOR

An AC^0-XOR circuit on variables $x_1, \ldots, x_n$ is an extension of AC^0 where gates at the bottom layer of the circuit can compute parity functions of arbitrary arity over $x_1, \ldots, x_n$. The size of an AC^0-XOR circuit is measured by the number of gates.

We first establish HM Frontier A in Section 4.3.1. Next, in Section 4.3.2, we show that a well-known technique for proving $\mathsf{AC}^0[2]$ lower bounds above the magnification threshold, i.e. the Razborov-Smolensky polynomial approximation method, localises and thus cannot be adapted to achieve magnification using Item A1.

4.3.1 Establishing HM Frontier A

Items A2 [RR97] and A3 [Raz87, Smo87] follow from known results. We start with the magnification theorem in Item A1.

Theorem 4.13. *There exists a constant $c_0 \geq 0$, such that for every large enough $c \geq 1$*

$$\mathsf{MKtP}[\log^c N, \log^c N + c_0 \log N] \notin \mathsf{AC}^0\text{-}\mathsf{XOR}[N^{1.01}] \implies \mathsf{EXP} \not\subseteq \mathsf{NC}^1$$

Before diving into the proof, we need the following structural result about NC^1-circuits.

Lemma 4.14 (cf. [IMP12], Lemma 8.1 in [AHM+08])**.** *For every sequence of NC^1-circuits $\{C_n\}_{n\in\mathbb{N}}$ and for every $d \geq 2$, there exists a sequence $\{D_n\}$ of depth-d AC^0-circuits of size $2^{n^{O(1/d-1)}}$, such that for each $n \in \mathbb{N}$, D_n computes the same function as C_n.*

Proof of Thoerem 4.13. We show the contrapositive of this result. Assume that $\mathsf{EXP} \subseteq \mathsf{NC}^1$. Consider the following language L which consists of input strings of the form $(a, 1^k, (i_1, b_1), \ldots, (i_r, b_r))$ (where $a, k \in \mathbb{N}$, a is encoded in binary and $b_j \in \{0, 1\}$), if for each $1 \leq j \leq r$, i_j is a string of length $\lceil \log a \rceil$ and there exists a string z of length a such that $\mathsf{Kt}(z) \leq k$ and $z_{i_j} = b_j$, for every index j.

[OPS19] show that $L \in \mathsf{EXP}$ and further, also show that for every large enough N and $c \geq 1$, $\mathsf{MKtP}[\log^c N, \log^c N + c_0 \log N]$ can be computed by a sequence of L-oracle circuits $\{D_N\}$ of the following form : The bottom layer has only XOR gates of arbitrary fan-in over the inputs, the middle layer has L-oracle gates of fan-in $m = O(\log^c N)$ and the output gate is an AND-gate of fan-in $O(N)$.

Under the assumption that $\mathsf{EXP} \subseteq \mathsf{NC}^1$, L can be computed by a sequence of polynomial-sized NC^1-circuits $\{C_n\}_{n\in\mathbb{N}}$. Using Lemma 4.14, for every $d \in \mathbb{N}$ and $m = O(\log^c N)$, C_m can be computed by an AC^0-circuit of size $2^{m^{1/d}}$, which is of size at most $N^{0.01}$ if d is picked to be large enough. Replacing each occurrence of the L_m-oracle with its equivalent depth-d AC^0 circuit, we obtain an AC^0-XOR circuit of superlinear size and depth $d + 2$ which computes $\mathsf{MKtP}[\log^c N, \log^c N + c_0 \log N]$. □

We next prove that even with a much larger gap, MKtP is not in AC^0. This in turn establishes HM Frontier A4.

Theorem 4.15. *For any $d = d(N)$, $\mathsf{MKtP}\left[d \cdot \widetilde{O}(\log^3 N), \frac{N}{\omega(\log^d N)}\right] \notin \mathsf{AC}^0_d$.*

We use random restrictions to show that $\mathsf{MKtP}[s_1, s_2]$ is not in AC^0, where $s_1 = d \cdot \widetilde{O}(\log^3 N)$ and $s_2 = N/\omega(\log^d N)$. The proof technique is similar to that of the exponential size AC^0 lower bound for MCSP in [CKLM19]. However, a crucial difference here is that we optimise on the Kt complexity of the YES instances instead of the size of the lower bound.

The idea is that when any polynomial sized constant-depth circuit C over N inputs is hit by a random restriction $\rho \sim \mathcal{R}_p$, it collapses to a shallow decision tree with high probability (see Lemma 4.9). This would allow us to take a fixed restriction ρ and either complete it into a YES instance z^Y of Kt complexity at most $s_1(N)$, or to a NO instance z^N of Kt complexity at least $s_2(N)$, if the number of unrestricted variables is large enough. However, the simplified circuit $C|_\rho$ now depends on very few variables in $\rho^{-1}(*)$, and thus $C|_\rho$ can't distinguish between z^Y and z^N, if they are suitably defined. This means that C_N cannot compute $\mathsf{MKtP}[s_1, s_2]$.

The main issue with this approach is that the restrictions we use to simplify C would restrict a large fraction of the N variables and a typical random restriction cannot be completed to get an instance with low Kt complexity. We thus use pseudorandom restrictions which can be computed from a small number of random bits by efficient algorithms.

Trevisan and Xue [TX13] showed that there exists a pseudorandom restriction $\mathcal{D}$ samplable using seeds of length $\mathsf{poly}(\log N)$, which collapses every polynomial-sized DNF (or a CNF) to a shallow decision tree with high probability. Moreover, the expected fraction of unrestricted variables is at least $p = \Omega(1/\log N)$. Under the restriction ρ obtained by composing d independently sampled pseudorandom restrictions, every polynomial sized constant-depth circuit C_N can be collapsed to a constant function, while leaving at least p^d many variables unrestricted.

Since the seed length for generating ρ is $d \cdot \mathsf{poly}(\log N)$, for the instance w^Y generated from the distribution $0^N \circ \rho$, $\mathsf{Kt}(w^Y) \leq \mathsf{poly}(\log N)$. On the other hand, we see that $C|_\rho$ cannot distinguish between $w^N = U_N \circ \rho$ (variables unrestricted by ρ are set by a uniform distribution) and $0^N \circ \rho$, since $C|_\rho$ is a constant function and does not depend on any of the unrestricted variables. In particular, if ρ leaves at least $N/O(\log^d N)$ many variables unrestricted, w^N has high Kt complexity and thus, C_N cannot compute $\mathsf{MKtP}[s_1, s_2]$.

Remark 4.16. *Notice that Theorem 4.15 is only meaningful when $d = o(\log N/\log\log N)$, because otherwise the gap for the promise problem is undefined.*

The rest of the section is devoted to the formal proof of Theorem 4.15. Trevisan and Xue stated a derandomised version of Håstad's switching lemma, i.e. showed that every pseudorandom distribution that fools large enough CNFs, generates pseudorandom restrictions under which small-width CNFs collapse to shallow decision trees with high probability. To state this formally, define a w-width CNF/DNF as one which has at most w variables in each clause.

Lemma 4.17 (Derandomised Switching Lemma [TX13] (Lemma 7))**.** *Let F be an S-clause w-width CNF over n variables, $p = 2^{-q}$ be a positive parameter for some $q \in \mathbb{N}$ and $\mathcal{D}$ be a distribution over $\{0,1\}^{(q+1)n}$ that ε_0-fools all $S \cdot 2^{w(q+1)}$-clause CNFs. Then, for every integer $t > 0$,*

$$\Pr_{z\sim\mathcal{D}}\{\mathsf{DT}_{depth}(F|_{\rho_z}) > t\} \leq 2^{t+w+1} \cdot (5pw)^t + \varepsilon_0 \cdot 2^{(t+1)(2w+\log S)}$$

We also need the following fact.

Fact 4.18 (Corollary 55 in [Tal17b])**.** *Let D be a distribution over $\{0,1\}^n$ that ε-fools CNFs of size S. For any $t \in \mathbb{N}$, let D^t be the distribution over $\{0,1\}^{n\cdot t}$ sampled by concatenating t independent samples from D. Then, D^t $(t\cdot\varepsilon)$-fools CNFs of size S on nt variables.*

We first show a version of derandomised switching lemma for AC^0.

Theorem 4.19 (Based on Theorem 11 in [TX13] and Theorem 56 in [Tal17b])**.** *Let C be a circuit of size S and depth d over N variables. Let $p = 2^{-q}$ for some $q \in \mathbb{N}$ and $t \in \mathbb{N}$ be a parameter so that $p < 1/64t$. Assume that there exists a polynomial-time pseudorandom generator $G : \{0,1\}^r \to \{0,1\}^{(q+1)N}$ that ε_0-fools CNFs of size $S \cdot 2^t \cdot 2^{t(q+1)}$. Then, there exists a distribution over restrictions $\mathcal{D}$ samplable by a generator $\mathcal{G}_0$ having seed length $d \cdot r$ which runs in polynomial time, such that*

1. $$\Pr_{\rho\sim\mathcal{D}}\{\mathsf{DT}_{depth}(C|_\rho) > t\} \leq S \cdot \left(2^{1-t} + \varepsilon_0 \cdot 2^{(t+1)(3t+\log S)}\right)$$
2. *For every $0 < \delta \leq 1$, with probability at least $(1 - N(\delta + \varepsilon_0 d))$, the number of unrestricted inputs is at least $\frac{Np^{d-1}}{64\log(1/\delta)}$.*

Proof of Theorem 4.19. The proof structure is very similar to that of [Tal17b] (cf. Theorem 56). W.l.o.g. we consider C as a $(d+1)$-depth circuit C' over n inputs where the fan-in of the bottom layer is 1 (can do this by adding a dummy layer of at most $S \cdot n$ many gates of fan-in 1). For each $1 \leq i \leq d$, let S_i be the number of gates in layer i of the original circuit C.

We construct $\mathcal{G}_0$ by sampling d independent pseudorandom restrictions using G. More formally, the generator $\mathcal{G}_0$ takes input $w = w_1, \ldots, w_d$, where each w_i is picked uniformly at random from $\{0,1\}^r$. It then outputs the restriction $\mathcal{G}_o(w) = \rho$ defined as $\rho = \rho_1 \circ \cdots \circ \rho_d$, where each ρ_i is independently sampled using G on the seed w_i. Clearly, the seed length of $\mathcal{G}_0$ is $d \cdot r$ and it runs in polynomial time since G does.

The proof proceeds by establishing the invariant that at the end of each iteration, C' can be simplified into a circuit whose depth reduces by 1, such that in the new circuit, every gate in layer 2 has a fan-in of at most $S \cdot 2^t$ (computes a CNF/DNF of size at most $S \cdot 2^t$) and the bottom fan-in is at most t. This can be observed by an inductive argument.

First Iteration: In the first iteration, we set $p = 1/64$ and $w = 1$ for the application of Lemma 4.17. We sample ρ_1 by computing $G(w_1)$ for a uniformly random seed w_1 and define ρ_1 by its first $(6+1)N = 7N$ bits. The probability that there exists a gate in layer 2 of $C'|_{\rho_1}$ that cannot be computed by a decision tree of depth at most t is at most $S_1 \cdot \left(2^{t+2} \cdot (5/64)^t + \varepsilon_0 \cdot 2^{(t+1)(2+\log S)}\right)$.

In other words, in the complement event, each gate at layer 2 in $C'|_{\rho_1}$ can be computed by a depth-t decision tree and thus, can be written as a t-width DNF (or a CNF)of size at most 2^t. This leads to the collapse of this layer into the one above it and yields a depth d-circuit, such that the fan-in of the gates in layer 2 of the new circuit is at most $S \cdot 2^t$, and in layer 1, is at most t.

The other $d-1$ iterations: For any other iteration $2 \leq i \leq d$, we set $p = 2^{-q}$ and $w = t$ for the application of Lemma 4.17. We first generate $\rho_i = G(w_i)$ and use the invariant from the $(i-1)^{\text{th}}$ iteration, to observe that the probability that there exists a gate at layer 2 in $C'|_{\rho_1,\ldots,\rho_i}$ that cannot be computed by a decision tree of depth at most t is at most $S_i \cdot \left(2^{2t+1} \cdot (5pt)^t + \varepsilon_0 \cdot 2^{(t+1)(2t+\log(S \cdot 2^t)}\right)$. In the complement event, each gate in layer 2 in $C'|_{\rho_1,\ldots,\rho_i}$ can be computed by an t-width DNF (or CNF) of size 2^t, and we collapse this layer into the one above it (reducing the depth of $C'|_{\rho_1,\ldots,\rho_i}$ by 1). The resulting circuit maintains the invariant that every gate in layer 2 has fan-in at most $S \cdot 2^t$ and the bottom fan-in is at most t.

At the end of d iterations, we use a union bound to observe that the probability that $\mathsf{DT}_{\text{depth}}(C|_\rho)$ is greater than t

$$
\begin{aligned}
&\leq \left(\sum_{i=1}^{d} S_i\right) \cdot \left(2^{2t+1} \cdot \max\{(5/64)^t, (5pt)^t\} + \varepsilon_0 \cdot 2^{(t+1)(2t+\log(S \cdot 2^t))}\right) \\
&\leq S \cdot \left(2^{2t+1} \cdot (5/64)^t + \varepsilon_0 \cdot 2^{(t+1)(2t+\log(S \cdot 2^t))}\right) \\
&\leq S \cdot \left(2^{1-t} + \varepsilon_0 \cdot 2^{(t+1)(3t+\log S)}\right)
\end{aligned}
$$

This shows Item 1 of the theorem.

To prove Item 2 of the theorem, we divide $[N]$ into B blocks of size k each, i.e. $B = \lfloor N/k \rfloor$, for k to be fixed later. The idea is to show that every block has at least one unrestricted variable with high probability.

Fix any block T_j for $1 \leq j \leq B$. [Tal17b] constructs a CNF of size at most N that takes $\rho_1, \ldots, \rho_d \in \{0,1\}^{d \cdot N(q+1)}$ as input and checks if every variable in T_j is restricted by their composition ρ. Using Fact 4.18 with the fact that G ε_0-fools CNFs of size $S \cdot 2^t \cdot 2^{t(q+1)}$, observe that $\mathcal{G}_0$ $(\varepsilon_0 d)$-fools CNFs of size $S \cdot 2^t \cdot 2^{t(q+1)}$ on $d(q+1)N$ variables.

Thus, the probability that all the variables in T_j are restricted by ρ is at most $\left(1 - p^{d-1}/64\right)^k + d\varepsilon_0$, where the first term in the expression is the probability that a truly random restriction fixes all the variables in T_j. For any $0 < \delta < 1$, by picking $k = 64 \log(1/\delta)/p^{d-1}$, this probability is at most $\delta + d\varepsilon_0$. Using a union bound over all the blocks, we see that each block has at least one unrestricted variable with probability at least $1 - (\lfloor N/k \rfloor \cdot (\delta + \varepsilon_0 d))$. In other words, with probability at least $(1 - N(\delta + \varepsilon_0 d))$, $|\rho^{-1}(*)| \geq \frac{Np^{d-1}}{64 \log(1/\delta)}$. □

Remark 4.20. *The proof of Theorem 4.15 is marginally different to that of [Tal17b] in a couple of ways: (1) we apply the derandomised switching lemma d times (instead of $d-1$) in order to turn depth-d circuits into shallow decision trees, and (2) for the application of constructing a PRG for AC^0, Tal (and also [TX13]) uses G to generate a pseudorandom selection (i.e. a string in $\{0,1\}^{qn}$) and set the bits of the restriction using truly random bits, whereas in our case we generate the entire restriction using G.*

Corollary 4.21. *Let C be an N-variate circuit of size S and depth d. There exists a restriction $\rho \in \{0,1,*\}^N$ such that*

1. *The decision tree depth of $C|_\rho$ is at most $t = 2 \log S$.*
2. *The number of unrestricted variables in ρ is at least $\frac{N}{O(\log^d S)}$.*
3. $\mathsf{Kt}(\rho) \leq d \cdot \widetilde{O}(\log^3 S)$.

For the proof, we need the following instantiation of a pseudorandom generator which fools DNFs/CNFs from [Tal17b].[6]

Lemma 4.22 (Theorem 52 in [Tal17b])**.** *There exists a polynomial time pseudorandom generator of seed length $\widetilde{O}(\log S \cdot (\log(S/\varepsilon_0)))$ which ε_0-fools all DNFs (or CNFs) of size S on n variables.*

[6] [Tal17b] uses a small-bias generator (see Definition 4.3) for this purpose.

Proof of Corollary 4.21. We apply Theorem 4.19 with parameters $t = 2\log S$ and $p = 1/65t$. To do so, we first use Lemma 4.22 with $\varepsilon_0 = 1/2^{15t^2}$ to give us a pseudorandom generator G which fools DNFs (or CNFs) of size $S \cdot 2^t \cdot 2^{t(q+1)}$, using a seed of length $r = \widetilde{O}(\log^3 S)$.

For these parameters, Theorem 4.19 shows that for restriction ρ sampled from $\mathcal{G}_0$, the probability that $\mathsf{DT}_{\mathrm{depth}}(C|_\rho)$ is greater than t is at most $1/4$. Further, for $\delta = \frac{3}{4N}$, $|\rho^{-1}(*)| < \frac{Np^{d-1}}{64\log(1/\delta)}$ with probability less than $1/4$. Thus, we see that there exists a restriction ρ generated by $\mathcal{G}_0$ such that the decision tree depth of $C|_\rho$ is at most t and the number of unrestricted variables is at least $\frac{N}{O(\log^d S)}$. This proves items 1 and 2.

To prove item 3, we see that the machine which computes the string that defines ρ has description length $O(dr) + O(1) = d \cdot \widetilde{O}(\log^3 S)$ (as we need to fix the random seed that generates ρ) and runs in time $\mathsf{poly}(N, r)$. Thus, the Kt complexity of ρ is at most $d \cdot \widetilde{O}(\log^3 S)$. □

We also make use of the following proposition.

Proposition 4.23 (Proposition 8 in [OPS19])**.** *Let B be an algorithm that runs in time $T(N)$ over inputs of length N. Then, for every $w \in \{0,1\}^N$, we have $\mathsf{Kt}(B(w)) \leq \mathsf{Kt}(w) + \log(T(N)) + O(1)$.*

Proof of Theorem 4.15. Let $s_1(N) = d \cdot \widetilde{O}(\log^3 N)$ and $s_2(N) = N/\omega(\log^d N)$. Assume that there exists a circuit C of size at most $S = \mathsf{poly}(N)$ and depth d that computes $\mathsf{MKtP}[d \cdot \widetilde{O}(\log^3 N), N/\omega(\log^d N)]$. We define a YES instance w^Y and a NO instance w^N such that $C(w^Y) = C(w^N)$ and this contradicts the fact that C computes $\mathsf{MKtP}[s_1(N), s_2(N)]$. Let $\rho \in \{0,1,*\}$ be a restriction given by Corollary 4.21.

- Define $w^Y \in \{0,1\}^N$ as $w^Y = 0^N \circ \rho$, i.e. all the variables unrestricted by ρ are set to 0. Next, observe that given ρ as input, there exists an algorithm which runs in polynomial time which outputs w^Y. In particular, this means that the $\mathsf{Kt}(w^Y) \leq \mathsf{Kt}(\rho) + a \cdot \log N$ from Proposition 4.23, for some universal constant $a \geq 0$. Since this is at most $s_1(N)$ (by choosing a large enough constant), we see that w^Y is a YES instance.

- Let $W \subseteq \rho^{-1}(*)$ be the set of variables of size at most $t = 2\log N$ which are queried by the smallest-depth decision tree of $C|_\rho$. The number of strings over $\rho^{-1}(*) \setminus W$ variables is at least $2^{N/O((\log^d N)) - 2\log N} > 2^{s_2(N)+1}$, where the latter quantity is the number of strings of length N with Kt complexity at most $s_2(N)$. In particular, there exists a string $h \in \{0,1\}^{\rho^{-1}(*)\setminus W}$ such that any string of length N which agrees with it has Kt complexity larger than $s_2(N)$. The string w^N is extended from ρ by additionally fixing the unrestricted variables in W to be 0, and setting the remaining

unrestricted variables in $\rho^{-1}(*) \setminus W$ to h. Clearly, w^N has Kt complexity at least $s_2(N)$.

Finally, $C(w^Y) = C(w^N)$, as w^Y and w^N agree on all inputs in $W \cup \rho^{-1}(\{0,1\})$, and C restricted by ρ depends only on W. □

4.3.2 Locality Barrier for $\mathsf{AC}^0[2]$-Lower Bounds Above the Threshold in HM Frontier A

In this section we observe that the Razborov-Smolensky [Raz87, Smo87] lower bound technique can be "localised".

Recall that the Razborov-Smolensky lower bound technique consists of two parts. The first part shows that $\mathsf{Majority}$ cannot be well-approximated by low-degree polynomials. In more detail, [Smo87] showed that no polynomial in $\mathbb{F}_2[x_1, \ldots, x_N]$ of degree at most $O(\sqrt{N})$ can agree with $\mathsf{Majority}$ on more than 4/5-fraction of the inputs.

Lemma 4.24 ([Smo87])**.** *There exists $c > 0$, such that for any $p \in \mathbb{F}_2[x_1, \ldots, x_N]$ of degree $c \cdot \sqrt{N}$*

$$\Pr_{x \in \{0,1\}^N}\{p(x) \neq \mathsf{Majority}(x)\} > 1/5$$

The second part shows that $\mathsf{AC}^0[2]$ circuits can be well-approximated by low-degree polynomials. A combination of these two proves the lower bound. We localise the second part by extending the result to $\mathsf{AC}^0[2]$ circuits with oracles having constrained fan-in. In particular, we show that -

Lemma 4.25. *For any $\mathsf{AC}^0[2]$ oracle circuit C of polynomial size and depth d, where the fan-in of the oracle gates in the i^{th}-level is at most O_i, there exists a polynomial $p \in \mathbb{F}_2[x_1, \ldots, x_N]$ of degree at most $O(\log^{d-1} N) \cdot \prod_{j=1}^{d} O_j$ such that $\Pr_{x \in \{0,1\}^N}\{p(x) \neq C(x)\} \leq 1/5$.*

The proof requires the notion of a low degree ε-error *probabilistic polynomial*. Recall from [Raz87, Smo87] that a probabilistic polynomial from $\mathbb{F}_2[x_1, \ldots, x_N]$ of degree at most D is a randomised multilinear polynomial $\mathcal{P}$ sampled from some distribution over polynomials in $\mathbb{F}_2[x_1, \ldots, x_n]$ of degree at most D. We say that $\mathcal{P}$ is an ε-error probabilistic polynomial for a function $f : \{0,1\}^n \to \{0,1\}$ if for each $x \in \{0,1\}^n$, we have $\Pr_{\mathcal{P}}\{\mathcal{P}(x) \neq f(x)\} \leq \varepsilon$.

Proof of Lemma 4.25. The proof is similar to the inductive probabilistic polynomial construction by [KS18] (cf. Lemma 3.6), which is a direct strengthening of [Raz87]. We start from the bottom layer, and work our way upwards to the output gate, while converting

the gates in each layer of C into a low degree probabilistic polynomial which approximates the gate well.

For the base case, each variable x_i can be trivially represented by a degree 1 polynomial. Next, for any $0 < i < d-1$, consider layer i of the circuit C. Assume that every sub-circuit rooted at a gate in level $i-1$ is approximated up to an error of $1/\mathsf{poly}(N)$ by a probabilistic polynomial on N variables and degree $O(\log^{i-1} N) \cdot \prod_{j=1}^{i-1} O_j$. Now, every $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}$ or XOR gate can be approximated by a probabilistic polynomial of degree $O(\log N)$ over the gate inputs, up to an error $1/\mathsf{poly}(N)$ using standard ideas from [Raz87]. Every oracle gate $\mathcal{O}$ can be represented by a polynomial by simply taking the multilinear extension of $\mathcal{O}$ of degree at most O_i, since the fan-in of the oracle gate in this layer is at most O_i. In particular, every gate in this layer can be represented by a probabilistic polynomial of degree at most $\max\{O(\log N), O_i\}$ up to error $1/\mathsf{poly}(N)$. Composing this polynomial with the polynomials from the $(i-1)^{\text{th}}$ layer, we see that each gate in this layer can be approximated by an N-variate probabilistic polynomial of degree at most $O(\log^i N) \cdot \prod_{j=1}^{i} O_j$.

Thus, each gate in layer $d-1$ can be represented by a probabilistic polynomial of degree at most $O(\log^{d-1} N) \cdot \prod_{j=1}^{d-1} O_j$. Finally, the output gate of C can be represented by a probabilistic polynomial of degree at most $\max\{O(1), O_d\}$.[7] By composing it with the polynomials for the gates in layer $d-1$, we get a probabilistic polynomial $\mathcal{P}$ on N variables and degree at most $O(\log^{d-1} N) \cdot \prod_{j=1}^{d} O_j$ that approximates C up to an error $1/5$. Using an averaging argument over $\mathcal{P}$, there exists a (deterministic) polynomial p of the same degree that computes C on at least a $4/5$-fraction of the inputs. □

The main result of this section is the following instantiation of the locality barrier for HM Frontier A.

Theorem 4.26 (Locality Barrier for HM Frontier A)**.** *The following results hold :*

(A1$^{\mathcal{O}}$) *(Oracle Circuit Construction from Magnification) There exists a constant $c \geq 0$ and an Oracle $\mathcal{O}$, such that for every large enough $d > 1$, $\mathsf{MKtP}[(\log^d N), (\log^d N) + c \log N] \in \mathsf{AND}\text{-}\mathcal{O}\text{-}\mathsf{XOR}[N^{1.01}]$, where each oracle gate has fan-in $O(\log^d N)$.*

(A3$^{\mathcal{O}}$) *(Extension of Lower Bound Techniques) Let $O_1, \ldots, O_d \in \mathbb{N}$ such that $\prod_{i=1}^{d} O_i \leq \frac{\sqrt{N}}{\omega(\log^{d-1} N)}$. Then, $\mathsf{Majority}$ on N inputs cannot be computed by any $\mathsf{AC}^0[2]$ oracle circuit of polynomial size and depth d, where any arbitrary oracle gates of fan-in O_i can be used in the i^{th} layer.*

[7] As in [KS18], if the output gate of C is an $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}$ or XOR gate, we can approximate it by a probabilistic polynomial of constant degree up to an error of $1/10$.

Proof. The proof of the first item follows from Theorem 4.13. For the second item, towards a contradiction, let C be an $\mathsf{AC}^0[2]$ oracle circuit having n inputs, polynomial size and depth d, which satisfies the constraints for oracle fan-in in the item 2 and computes $\mathsf{Majority}$. From Lemma 4.25 we see that there exists a polynomial of degree $O(\log^{d-1} N) \cdot \prod_{j=1}^{d} O_d < o(\sqrt{N})$ that approximates $\mathsf{Majority}$ up to an error of at most $1/5$, which contradicts Lemma 4.24. □

An incomparable lower bound can also be obtained for $(\mathrm{A3}^{\mathcal{O}})$ by using the lower bound for $\mathsf{AC}^0[2]$ interactive compression games from [OS15].

Proposition 4.27 (Corollary 5.3 in [OS15])**.** $\mathsf{Majority}_n$ *cannot be computed by* $\mathsf{AC}^0[2]$ *circuits of polynomial size with arbitrary oracle gates, if the total fan-in of the oracle gates is* $N/(\log N)^{\omega(1)}$.

4.4 HM Frontier for Gap MCSP and Formula-XOR

Recall that $\mathsf{Formula\text{-}XOR}[s]$ is the class of De Morgan formulas on variables $x_1, \ldots, x_n$ with at most s leaves, where each leaf is a parity function of arbitrary arity over $x_1, \ldots, x_n$.

We first establish HM Frontier B in Section 4.4.1. Following this, we show that known lower bound techniques by [Tal17a] and [KKL+20] against $\mathsf{Formula\text{-}XOR}$ localise in Section 4.4.2.

4.4.1 Establishing HM Frontier B

In this section we establish the HM Frontier B. Items B2 [RR97], B3(a) [Tal17a], B3(b) [KKL+20] and B4 [HS17, OPS19] follow from known results. We show the hardness magnification result in Item B1 here.

In fact, we provide two different proofs for Item B1, one based on [OS18a] and the other based on [CJW19]. In both proofs, magnification is achieved by constructing a low fan-in oracle circuit for MCSP. We present the first approach here and the second approach in Section 4.6.1, as it is obtained as a by-product of proving Item D1.

Theorem 4.28 (HM for worst-case MCSP via reductions)**.** *Suppose that there exists* $\varepsilon > 0$*, such that* $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula\text{-}XOR}[N^{1+\varepsilon}]$*. Then* $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$.

The most interesting part of the proof of Theorem 4.28 is that we get a *conditional* oracle circuit construction, assuming that $\mathsf{QP} \subseteq \mathsf{P/poly}$. Although, the oracle construction (Theorem 4.45) from Section 4.6.1 is *unconditional*, the proof in this section illustrates a potentially more applicable approach where the assumption in the contrapositive (i.e.

$\mathsf{NQP} \subseteq \mathsf{NC}^1$) is also needed for the construction of the oracle circuits which imply magnification.

[OS18a] showed hardness magnification for approximate MCSP. Theorem 4.28 uses their result to show a similar phenomenon for the worst-case version of MCSP.

A natural approach to establish a reduction from worst-case gap MCSP to approximate MCSP is by using error-correcting codes. Error-correcting codes map a Boolean function which is hard in the worst-case to one which is hard on average. One issue with using codes is that, if the function is easy to begin with (i.e. can be computed by a small-sized circuit), the generated codeword has no guarantee about being easily computable. We show that this property holds under the assumption that $\mathsf{QP} \subseteq \mathsf{P/poly}$ (which follows from $\mathsf{NQP} \subseteq \mathsf{NC}^1$).[8]

We use the following error-correcting code.

Theorem 4.29 (Explicit linear error-correcting codes [Jus72, SS96])**.** *There exists a sequence $\{E_N\}$ of error-correcting codes $E_N \colon \{0,1\}^N \to \{0,1\}^{M(N)}$ with the following properties:*

- *$E_N(x)$ can be computed by a uniform deterministic algorithm running in time $\mathsf{poly}(N)$.*
- *$M(N) = b \cdot N$ for a fixed $b \geq 1$.*
- *There exists a constant $\delta > 0$ such that any codeword $E_N(x) \in \{0,1\}^{M(N)}$ that is corrupted on at most a δ-fraction of coordinates can be uniquely decoded to x by a uniform deterministic algorithm D running in time $\mathsf{poly}(M(N))$.*
- *Each output bit is computed by a parity function: for each input length $N \geq 1$ and for each coordinate $i \in [M(N)]$, there exists a set $S_{N,i} \subseteq [N]$ such that for every $x \in \{0,1\}^N$,*

$$E_N(x)_i = \bigoplus_{j \in S_{N,i}} x_j.$$

We need the following result from [OS18a] for the proof.

Lemma 4.30 (Lemma 16 from [OS18a])**.** *If there exists $\varepsilon > 0$ and constant $\delta > 0$ such that $\mathsf{MCSP}[(2^{\sqrt{n}}, 0), (2^{\sqrt{n}}, \delta)] \notin \mathsf{Formula\text{-}XOR}[N^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.*

[8] The proof of Theorem 4.28 shows a conditional reduction from worst-case MCSP to approximate MCSP. Recall Section 3.4 for connections between such reductions and hardness of learning from worst-case assumptions.

Proof of Theorem 4.28. Towards proving the contrapositive, assume that $\mathsf{NQP} \subseteq \mathsf{NC}^1$. For any $N = 2^n$, let $w \in \{0,1\}^N$ be the input instance to $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$. We first show a reduction from $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ to $\mathsf{MCSP}[(2^{\sqrt{n}}, 0), (2^{\sqrt{n}}, \delta)]$, for $\delta > 0$ given by Theorem 4.29. The reduction maps w to $E_N(w) = z$, where E_N is the encoder given by the code in Theorem 4.29 and $z \in \{0,1\}^{M(N)}$. Let $m = \log M(N)$.

We first show that if $\mathsf{fn}(w)$ can be computed by a circuit C_n of size $2^{n^{1/3}}$, then $\mathsf{fn}(z)$ can be computed by a circuit of size $2^{\sqrt{m}}$ over m inputs. Define function F which takes as inputs an encoding of C_n and $i \in \{0,1\}^m$, and outputs the i^{th} bit of $E_N(w)$. The size of the input to F is $2^{O(n^{1/3})}$ and $E_N(w)$ runs in time $2^{O(n)}$. Since $\mathsf{NQP} \subseteq \mathsf{NC}^1$, we see that $\mathsf{QP} \subseteq \mathsf{P/poly}$ and thus, F can be computed by a polynomial-sized circuit. In other words, there exists a circuit F' of size $2^{O(n^{1/3})}$ into which we can hard-wire the description of the circuit C_n, such that on input i, $F'(C_n, i)$ outputs $\mathsf{fn}(z)(i)$. Clearly, $\mathsf{fn}(z)$ can be computed by a circuit of size $2^{O(n^{1/3})} \leq 2^{\sqrt{m}}$.

For the other direction, suppose $\mathsf{fn}(z)$ can be computed by a circuit B_m of size $2^{\sqrt{m}}$ on at least $(1-\delta)$-fraction of the inputs of length m. Define G which takes B_m and $j \in \{0,1\}^n$ as inputs and outputs the j^{th} bit of $D_N(\mathsf{tt}(B_m))$, where D_N is the decoder given by Theorem 4.29. Observe that G is computable in time $2^{O(n)}$ and thus, is in $\mathsf{QP} \subseteq \mathsf{P/poly}$. In other words, there exists a circuit G' of size $2^{O(\sqrt{m})} \leq 2^{n^{2/3}}$ into which we can hardwire the description of B_m, such that on input j, $G'(B_m, j)$ outputs $\mathsf{fn}(w)(j)$.

Finally, we use Lemma 4.30 along with the assumption that $\mathsf{NQP} \subseteq \mathsf{NC}^1$, to obtain that $\mathsf{MCSP}[(2^{\sqrt{m}}, 0), (2^{\sqrt{m}}, \delta)] \in \mathsf{Formula}[N^{1+\varepsilon}]$, for every $\varepsilon > 0$. Since each output bit of E_N can be computed by an XOR gate, the reduction can be implemented by a linearly many XOR gates at the bottom. Combining the formula above with this reduction, we see that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \in \mathsf{Formula\text{-}XOR}[N^{1+\varepsilon}]$. □

4.4.2 Locality Barrier for Formula-XOR Lower Bounds Above the Threshold in HM Frontier B

This section captures an instantiation of the locality barrier for HM Frontier B. In this section we use the $\{-1,1\}$ realisation of the Boolean domain (recall that -1 represents True and 1 represents false). Define the Inner Product modulo 2 function, $\mathsf{InnerProduct}_n : \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ as $\mathsf{InnerProduct}(x,y) = (-1)^{\sum_{i=1}^{n}(1-x_i)(1-y_i)/4}$.

Theorem 4.31 (Locality Barrier for HM Frontier B)**.** *The following results hold.*

(B1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) *There exists an oracle $\mathcal{O}$, such that for any $\varepsilon > 0$, $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \in \mathsf{Formula}\text{-}\mathcal{O}\text{-}\mathsf{XOR}[N^{1.01}]$, where every oracle gate has fan-in at most N^{ε} and appears in the layer right above the XOR leaves.*

(B3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques - 1) *For any $\delta > 0$,* InnerProduct *over N inputs cannot be computed by $N^{2-2\delta}$-size* Formula-$\mathcal{O}$-XOR *circuits with at most $N^{2-3\delta}$ oracle gates of fan-in N^{δ} in the layer right above the* XOR *leaves, for any oracle $\mathcal{O}$.*

(B3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques - 2) *For every $\delta > 0$,* MCSP$[2^n/n^4]$ *cannot be computed by $N^{2-2\delta-o(1)}$-size* Formula-$\mathcal{O}$-XOR *circuits with at most $N^{2-3\delta-o(1)}$ oracle gates of fan-in N^{δ} in the layer right above the* XOR *leaves, for any oracle $\mathcal{O}$.*

To prove item 2 of Theorem 4.31, we adapt Tal's [Tal17a] lower bound for bipartite formulas,[9] for which we need the following results.

Lemma 4.32 ([Rei11, Tal17a])**.** *Let F be a De Morgan formula of size s which computes $f : \{-1,1\}^n \to \{1,1\}$. Then, there exists a multilinear polynomial p over $\mathbb{R}$ of degree $O(\sqrt{s})$, such that for every $x \in \{-1,1\}^n$, $p(x) \in [F(x) - 1/3, F(x) + 1/3]$.*

For any function $f : \{-1,1\}^n \to \{-1,1\}$, f is said to be ε-correlated with a parity $p_S(x) = \prod_{i \in S} x_i$, if $\left|\mathbb{E}_{x \in \{-1,1\}^n}[f(x) \cdot p_S(x)]\right| \geq \varepsilon$. Next, we show the following result on pointwise approximating formula-XOR circuits with oracles using polynomials.

Lemma 4.33. *For any $\delta > 0$, let $F(x_1, \ldots, x_n)$ be a* Formula-$\mathcal{O}$-XOR *circuit of size s, where every oracle $\mathcal{O}$ has fan-in at most n^{δ} and appears in the layer right above the* XOR *leaves. Then the following hold:*

1. *There exists a multilinear polynomial $p(x_1, \ldots, x_n)$ over $\mathbb{R}$ with at most $s^{O(\sqrt{s})} \cdot 2^{n^{\delta} \cdot O(\sqrt{s})}$ monomials such that for every $x \in \{-1,1\}^n$, $sign(p(x)) = F(x)$.*
2. *There exists a parity function $p_T(x_1, \ldots, x_n)$ which is at least $\left(\frac{1}{s^{O(\sqrt{s})} \cdot 2^{n^{\delta} O(\sqrt{s})}}\right)$-correlated with F.*

Proof. We assume that each oracle gate is on $\ell = n^{\delta}$ inputs. Let $t \leq s/n^{\delta}$ be the number of oracle gates in F. Let $p_1, \ldots, p_s$ be the leaves of F, where each p_i is an XOR gate over $x_1, \ldots, x_n$. Let $g_1, \ldots, g_t$ be oracle gates such that $g_i(x) = \mathcal{O}(p_{i1}(x), \ldots, p_{i\ell}(x))$, where $p_{ij} \in \{p_1, \ldots, p_s\}$ for every $i \in [t], j \in [\ell]$.

Let F' be a De Morgan formula obtained by replacing each oracle gate in F with a new variable z_i (for notational simplicity we assume that every leaf is an input to some oracle gate). We now use Lemma 4.32 on F' to get a t-variate polynomial $q(z)$ over $\mathbb{R}$ of

[9] A bipartite formula on variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ is a formula such that each leaf computes an arbitrary function in either $(x_1, \ldots, x_n)$ or $(y_1, \ldots, y_n)$. Formula-XOR circuits are a subset of bipartite formulas as one can always write $\oplus(x_1, \ldots, x_{2n})$ as the parity of $\oplus(x_1, \ldots, x_n)$ and $\oplus(x_{n+1}, \ldots, x_{2n})$.

degree $d = O(\sqrt{t})$, such that for every $z \in \{-1,1\}^t$, $\text{sign}(q(z)) = F'(z)$. Expanding $q(z)$ as a multilinear polynomial :

$$q(z) = \sum_{S \subseteq [t], |S| \leq d} \widehat{q}(S) \prod_{i \in S} z_i$$

To prove the first item we substitute z_i back with its corresponding oracle gate $g_i(x)$. For every $x \in \{-1,1\}^n$, we have

$$\begin{aligned}
F(x) &= \text{sign}\left(\sum_{S \subseteq [t], |S| \leq d} \widehat{q}(S) \prod_{i \in S} g_i(x) \right) \\
&= \text{sign}\left(\sum_{S \subseteq [t], |S| \leq d} \widehat{q}(S) \prod_{i \in S} \left(\sum_{U \subseteq [\ell]} \widehat{\mathcal{O}}(U) \prod_{j \in U} p_{ij}(x) \right) \right) \\
&= \text{sign}\left(\sum_{\substack{S \subseteq [t] \\ S = \{i_1, \ldots, i_{|S|}\} \\ |S| \leq d}} \sum_{U_{i_1}, \ldots, U_{i_{|S|}} \subseteq [\ell]} \widehat{q}(S) \cdot \left(\prod_{1 \leq k \leq |S|} \widehat{\mathcal{O}}(U_{i_k}) \prod_{j \in U_{i_k}} p_{i_k j}(x) \right) \right)
\end{aligned}$$

where the second equality uses the fact that any Boolean function on ℓ inputs can be represented by a multilinear polynomial of degree at most ℓ where each coefficient is between $[-1,1]$. Clearly the number of monomials is at most $s^{O(\sqrt{s})} \cdot 2^{n^\delta \cdot O(\sqrt{s})}$.

To prove the second item, firstly observe that for every $z \in \{-1,1\}^t$, $q(z) \cdot F'(z) \in [2/3, 4/3]$, because $|q(z) - F'(z)| \leq 1/3$ for every z. This also means that for the polynomial $r(x) = q(g_1(x), \ldots, g_t(x))$, $\mathbb{E}_{x \in \{-1,1\}^n}[r(x) \cdot F(x)] \geq 2/3$.

Given that $\widehat{q}(S) = \mathbb{E}_{z \in \{-1,1\}^t}\left[q(z) \prod_{i \in S} z_i\right]$, we see that $|\widehat{q}(S)| \leq 4/3$. We have

$$\begin{aligned}
2/3 &\leq \mathop{\mathbb{E}}_{x \in \{-1,1\}^n}[F(x) \cdot r(x)] \\
&= \mathop{\mathbb{E}}_{x \in \{-1,1\}^n}\left[F(x) \cdot \sum_{S \subseteq [t], |S| \leq d} \widehat{q}(S) \prod_{i \in S} g_i(x) \right] \\
&\leq \sum_{\substack{S \subseteq [t] \\ S = \{i_1, \ldots, i_{|S|}\} \\ |S| \leq d}} \sum_{U_{i_1}, \ldots, U_{i_{|S|}} \subseteq [\ell]} \widehat{q}(S) \cdot \prod_{1 \leq k \leq |S|} \widehat{\mathcal{O}}(U_{i_k}) \cdot \mathop{\mathbb{E}}_{x \in \{-1,1\}^n}\left[F(x) \cdot \prod_{1 \leq k \leq |S|} \prod_{j \in U_{i_k}} p_{i_k j}(x) \right]
\end{aligned}$$

Since, $|\widehat{q}(S)| \leq 4/3$ for every $S \subseteq [t]$ and $|\widehat{\mathcal{O}}(U)| \leq 1$ for every $U \subseteq [\ell]$, we see that there exists a set S of size at most d and sets $U_{i_1}, \ldots, U_{i_{|S|}}$ such that

$$\left| \mathbb{E}_{x \in \{-1,1\}^n}\left[F(x) \cdot \prod_{1 \leq k \leq |S|} \prod_{j \in U_{i_k}} p_{i_k j}(x) \right] \right| \geq \frac{1}{t^{O(\sqrt{t})} \cdot 2^{n^\delta O(\sqrt{t})}} \geq \frac{1}{s^{O(\sqrt{s})} \cdot 2^{n^\delta O(\sqrt{s})}}$$

Taking p_T be the parity of the parities defined by $p_T = \prod_{1 \leq k \leq |S|} \prod_{j \in U_{i_k}} p_{i_k j}(x)$, we see that p_T is $\frac{1}{s^{O(\sqrt{s})} \cdot 2^{n^\delta O(\sqrt{s})}}$-correlated with F. □

Proof Sketch of Items 1 and 2 of Theorem 4.31. The first item follows from an inspection of the proof of Theorem 4.45 in Section 4.6.1.[10]

The second item follows from Lemma 4.33. We observe that three different techniques used to show Formula-XOR lower bounds by [Tal17a] localise.

- Tal's proof can be extended to show that the sign rank[11] of any Formula-$\mathcal{O}$-XOR circuit F having the desired structure is at most the number of monomials in the polynomial p given by the first item of lemma 4.33. Since this is at most $s^{O(\sqrt{s})} \cdot 2^{n^\delta \cdot O(\sqrt{s})}$ and InnerProduct has a sign rank which is at least $2^{n/2}$ [For02], the lower bound follows.

- Tal's lower bound based on the discrepancy of a function also localises.[12] He shows that the discrepancy of F is at least a constant times the correlation of F with the parity f_T given by item 2 of Lemma 4.33, which is at least $\Omega\left(\frac{1}{s^{O(\sqrt{s})} \cdot 2^{n^\delta O(\sqrt{s})}}\right)$. On the other hand, the discrepancy of InnerProduct is at most $1/2^{n/2}$ (cf. Lemma 14.5 of [Juk12]), thus proving the given lower bound for InnerProduct.

- Finally, we observe that the lower bound technique of showing a high correlation of F with some parity f_T, in combination with the fact that InnerProduct has exactly $2^{-n/2}$-correlation with any parity also localises to give the same lower bound.[13]

□

Similar ideas can be used to localise the MCSP lower bound against sub-quadratic Formula-XOR circuits by [KKL+20]. The main idea behind their proof is the following lemma which is used to get PRGs for Formula-$\mathcal{G}$ from small-error PRGs for XOR-$\mathcal{G}$.

Lemma 4.34. *Let $\mathcal{G}$ be a class of functions over n bits. For any integer $s = s(N)$ and any $0 < \varepsilon < 1$, if a distribution $\mathcal{D}$ over $\{-1,1\}^N$ γ-fools the XOR of any $O(\sqrt{s}\log(1/\varepsilon))$ functions from $\mathcal{G}$, where $\gamma = \exp(O(\sqrt{s}\log s \cdot \log(1/\varepsilon)))$, then $\mathcal{D}$ also ε-fools Formula-$\mathcal{G}[s]$.*

[10] Note that Theorem 4.28 in Section 4.4.1 also gives us the same oracle circuit construction, under the assumption QP $\subseteq$ P/poly (although, with different oracles).

[11] For any matrix $A \in \{-1,+1\}^{m \times n}$, the *sign-rank* of A is defined as the least rank of a matrix $B \in \mathbb{R}^{m \times n}$ such that for every $(i,j) \in [m] \times [n]$, $B(i,j) \neq 0$ and $\mathrm{sign}(B_{i,j}) = A_{i,j}$. The sign-rank of any Boolean function $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$ is defined as the sign-rank of the $2^n \times 2^n$ matrix M_f, given as $M_f[x,y] = f(x,y)$.

[12] The *discrepancy* of any Boolean function $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$ is defined as $\frac{1}{2^{2n}} \max_{I,J \subseteq [2^n]} \left| \sum_{x \in I} \sum_{y \in J} M_f(x,y) \right|$.

[13] It is well-known that InnerProduct is a *bent* function, i.e. each Fourier coefficient is equal to $2^{-n/2}$ (see [O'D14]).

We use this framework to construct a PRG which ε-fools $\mathsf{Formula}\text{-}\mathcal{O}_{n^\delta}\text{-}\mathsf{XOR}[s(n)]$ circuits over n inputs.

Lemma 4.35. *For any integer $s(n)$, $0 < \varepsilon < 1$ and $\delta > 0$, any $\exp(O(-\sqrt{s}\log(1/\varepsilon) \cdot (\log s + n^\delta)))$-biased distribution ε-fools $\mathsf{Formula}\text{-}\mathcal{O}_{n^\delta}\text{-}\mathsf{XOR}[s]$.*

Proof. Let $\gamma' = \exp(O(-\sqrt{s}(\log s + n^\delta) \cdot \log(1/\varepsilon)))$. Next, let $\mathcal{G} = \mathcal{O}_{n^\delta}\text{-}\mathsf{XOR}$ over n inputs and $\mathcal{D}$ be any γ'-biased distribution from Definition 4.3. Finally, let $\ell = n^\delta$ and $t = O(\sqrt{s}\log(1/\varepsilon)))$. Let $g_1, \ldots, g_t$ be any functions from $\mathcal{G}$ such that $g_i(x) = \mathcal{O}(p_{i1}(x), \ldots, p_{i\ell}(x))$, where each p_{ij} is an XOR gate of arbitrary arity over the n inputs. Let $F(x) = \prod_{i=1}^{t} g_i(x)$.

For $\gamma = \gamma' \cdot 2^{t\ell} = \exp(O(-\sqrt{s}\log s \log(1/\varepsilon)))$, we show that $\mathcal{D}$ γ-fools the XOR of $O(\sqrt{s} \cdot \log(1/\varepsilon))$ many functions from $\mathcal{G}$. Using Lemma 4.34 with these parameter settings proves the result.

Since $\mathcal{O}_\ell$ can be represented by a multilinear polynomial of degree at most ℓ where each coefficient is between $[-1, 1]$, we have

$$\begin{aligned} F(x) &= \prod_{1\le k\le t} \mathcal{O}(p_{k1}(x), \ldots, p_{k\ell}(x)) = \prod_{1\le k\le t} \left(\sum_{S\subseteq[\ell]} \widehat{\mathcal{O}}(S) \left(\prod_{j\in S} p_{kj}(x) \right) \right) \\ &= \sum_{V_1,\ldots,V_t\subseteq[\ell]} \left(\prod_{1\le k\le t} \left(\widehat{\mathcal{O}}(V_k) \prod_{j\in V_k} p_{kj}(x) \right) \right) \end{aligned}$$

Now, we have

$$\begin{aligned} \mathbf{E}_{x\sim\mathcal{D}}[F(x)] &= \sum_{V_1,\ldots,V_t\subseteq[\ell]} \left(\prod_{1\le k\le t} \widehat{\mathcal{O}}(V_k) \right) \cdot \mathbf{E}_{x\sim\mathcal{D}} \left[\prod_{1\le k\le t} \prod_{j\in V_k} p_{kj}(x) \right] \\ &\le \sum_{V_1,\ldots,V_t\subseteq[\ell]} \left(\prod_{1\le k\le t} \widehat{\mathcal{O}}(V_k) \right) \cdot \left(\mathbf{E}_{x\sim U_n} \left[\prod_{1\le k\le t} \prod_{j\in V_k} p_{kj}(x) \right] + \gamma' \right) \\ &\le \mathbf{E}_{x\sim U_n} \left[\sum_{V_1,\ldots,V_t\subseteq[\ell]} \left(\prod_{1\le k\le t} \left(\widehat{\mathcal{O}}(V_k) \prod_{j\in V_k} p_{kj}(x) \right) \right) \right] \\ &\qquad + \sum_{V_1,\ldots,V_t\subseteq[\ell]} \gamma' \left(\prod_{1\le k\le t} \widehat{\mathcal{O}}(V_k) \right) \\ &\le \mathbf{E}_{x\sim U_n}[F(x)] + 2^{t\ell} \cdot \gamma' \end{aligned}$$

where the second inequality holds because $\mathcal{D}$ γ'-fools $\prod_{1\le k\le t} \prod_{j\in V_k} p_{kj}(x)$ which is a large parity and the final inequality holds because $\widehat{\mathcal{O}}(V_k) \le 1$ for every V_k. □

We also need the following result to show the MCSP lower bound.

Theorem 4.36 (Section 3 of [CKLM19]). *There exists a constant $c > 0$ such that the following holds. For any circuit class $\mathcal{C}$, suppose that $\mathsf{MCSP}[2^n/n^4]$ is computable in $\mathcal{C}[s]$, where $s = s(N)$. If there exists a psuedorandom generator $G : \{0,1\}^r \to \{0,1\}^N$ that $1/3$-fools $\mathcal{C}[s]$ such that the circuit complexity of every string in the output of G (seen as a truth table) is $T(N,s)$, then $T \geq \frac{N}{c \log N}$.*

Proof of Item 3 of Theorem 4.31. For any $s = s(N)$, Lemma 4.35 gives the construction of a PRG $G : \{0,1\}^r \to \{0,1\}^N$ that ε-fools $\mathsf{Formula}\text{-}\mathcal{O}_{N^\delta}\text{-}\mathsf{XOR}[s]$ where:

- $r = O(\sqrt{s}\log(1/\varepsilon) \cdot (\log s + N^\delta) + \log N)$.
- every string in the output of G has circuit complexity $T = O(r \cdot \mathsf{poly}(\log N))$ (Lemma 4.4).

Suppose that $\mathsf{MCSP}[2^n/n^4] \in \mathsf{Formula}\text{-}\mathcal{O}_{N^\delta}\text{-}\mathsf{XOR}[s]$. From Theorem 4.36, we have

$$O(\sqrt{s}\log(1/\varepsilon) \cdot (\log s + N^\delta) + \log N) \cdot \mathsf{poly}(\log N) \geq \frac{N}{c \log N}$$
$$\implies s \geq \Omega\left(\frac{N^{2-2\delta}}{\mathsf{poly}(\log N)}\right)$$

□

4.4.3 On Lower Bounds through Reductions

In this section we show that the locality barrier also rules out reductions from $\mathsf{InnerProduct}$ to $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$, that would imply stronger lower bounds via magnification.

More precisely, consider a reduction from $\mathsf{InnerProduct}_N$ to $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ of the following form: Suppose that there exists $\delta > 0$, such that the reduction is a sub-quadratic $\mathsf{Formula}\text{-}\mathcal{O}\text{-}\mathsf{XOR}$ circuit with a certain number of (possibly non-local) oracle gates above the XOR layer, where each oracle gate computes $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$, and on replacing each MCSP oracle with a $\mathsf{Formula}\text{-}\mathsf{XOR}$ circuit of size $N^{1.01}$, we get a $\mathsf{Formula}\text{-}\mathsf{XOR}$ circuit of total size $N^{2-2\delta}$ which computes $\mathsf{InnerProduct}$. The reduction implies that $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \notin \mathsf{Formula}\text{-}\mathsf{XOR}[N^{1.01}]$, since otherwise we would contradict [Tal17a].

All in all, such a reduction would show that $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ via magnification from Theorem 4.28. Unfortunately, Theorem 4.31 also rules out this possibility.

Theorem 4.37. *Let $\mathcal{O} = \mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$. For every $\delta > 0$, $\mathsf{InnerProduct}_N$ cannot be computed by $\mathsf{Formula}\text{-}\mathcal{O}\text{-}\mathsf{XOR}$-circuits with at most $N^{1-3\delta}$ $\mathcal{O}$-oracle gates above the XOR layer, such that replacing each oracle with a $\mathsf{Formula}\text{-}\mathsf{XOR}$ circuit of size $N^{1.01}$ would result in a $\mathsf{Formula}\text{-}\mathsf{XOR}$ circuit of total size at most $N^{2-2\delta}$.*

Proof. Suppose such a reduction exists for some $\delta > 0$. From Item 1 of Theorem 4.31, there exists a language L, such that for every $1 > \varepsilon > 0$, $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ can be computed by Formula-XOR oracle circuits of $N^{1.01}$ size, with $O(N)$-many L-oracle gates of fan-in N^{ε} in the layer above the XOR leaves.

Replacing the MCSP oracles in the reduction with these circuits, we get a Formula-$\mathcal{O}$-XOR circuit of size at most $N^{2-2\delta}$ (from the assumption in the theorem), with $O(N^{2-3\delta})$ oracles of fan-in N^{δ} right above the XOR layer. This contradicts Item 2 of Theorem 4.31.

One minor caveat for the reduction to hold is that the fan-in of the MCSP oracle gates in the reduction needs to be $B = \omega(N^{\delta})$, so that we can employ the oracle circuit construction for MCSP with an appropriate $1 > \varepsilon > 0$, i.e. ε is such that $B^{\varepsilon} = N^{\delta}$. □

Based on their respective locality barriers, analogous arguments rule out the possibility of establishing strong lower bounds using structured reductions in the other HM Frontiers as well.

4.5 HM Frontier for Gap MCSP and Almost-Formulas

Define a t-almost-formula of size s as a Boolean circuit with at most s AND, OR and NOT gates of fan-in at most 2, such that at most t of these gates have fan-out larger than 1. Almost-formulas can be thought of as a model which naturally interpolates between formulas (where all gates have fan-out at most 1) and circuits.

We first establish HM Frontier C in Section 4.5.1. Following this, in Section 4.5.2 we show that the lower bound for Parity against almost-formulas localises.

4.5.1 Establishing HM Frontier C

In this section, we establish HM Frontier C. We begin with the hardness magnification theorem in Item C1.

On one hand, [OPS19] show that if worst-case $\mathsf{MCSP}[2^{n^{o(1)}}, 2^{n^{o(1)}}]$ does not have super-linear sized circuits, then $\mathsf{NP} \not\subseteq \mathsf{P/poly}$. They proved this statement using a highly-efficient construction of *anti-checkers*, i.e. a bounded set of inputs which witnesses a lower bound for a hard function f against a circuit class. On the other hand, recall that $\mathsf{MCSP}[2^{n^{o(1)}}, 2^{n^{o(1)}}]$ has a near-*quadratic* formula size lower bound from [HS17]. Consequently, if one could extend this magnification theorem to work for formulas, $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ would follow. We take a step in this direction by showing that if worst-case $\mathsf{MCSP}[2^{n^{o(1)}}, 2^{n^{o(1)}}]$ cannot be computed by super-linear sized almost-formulas, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

Theorem 4.38 (Magnification for almost-formulas)**.** *If there exists $\varepsilon > 0$, such that* $\mathsf{MCSP}[2^{\sqrt{n}}/2n, 2^{\sqrt{n}}] \notin 2^{O(\sqrt{n})}\text{-}\mathsf{Almost\text{-}Formula}[N^{1+\varepsilon}]$*, then* $\mathsf{NP} \not\subseteq \mathsf{NC}^1$*.*

In Section 4.8.2, we rule out a large family of theorems which magnify worst-case MCSP lower bounds against super-linear formulas to strong lower bounds, including those based on the techniques of this section (or [OPS19]). Along with Theorem 4.38, this indicates that the usage of almost-formulas is unavoidable for magnification theorems based on such techniques.

Theorem 4.38 is proved using a highly-efficient construction of anti-checkers. Roughly speaking, an *anti-checker* is a bounded multi-set S of input instances associated with a hard function f such that every small circuit differs from f on some input in S. In more detail, [LY94] established that for any function f on n inputs that cannot be computed by circuits of size s, there exists a collection S_f containing $O(s)$ strings such that every circuit of size $s/2n$ disagrees with f on at least one input in S. [OPS19] show an anti-checker construction using highly efficient general circuits, assuming $\mathsf{NP} \subseteq \mathsf{P/poly}$. The main technical result of this section is an iterative construction of anti-checkers which builds on [OPS19], but ensures that very few gates in the circuit have fan-out larger than 1, assuming that $\mathsf{NP} \subseteq \mathsf{NC}^1$.

Lemma 4.39 (Anti-Checker construction)**.** *Suppose that* $\mathsf{NP} \subseteq \mathsf{NC}^1$*. Then, for every $\lambda > 0$, there exists a sequence of circuits $\{C_{2^n}\}_{n\in\mathbb{N}}$ of size $2^{n+O(n^\lambda)}$ which takes as input a truth table $\mathsf{tt}(f) \in \{0,1\}^N$, and outputs $2^{O(n^\lambda)}$ strings $S_f = \{y_1, \ldots, y_{2^{O(n^\lambda)}}\}$ along with the evaluations of f on these strings, $f(y_1), \ldots, f(y_{2^{O(n^\lambda)}})$, such that if f is hard for circuits of size 2^{n^λ}, S_f is an anti-checker for f.*

Moreover, each $y_i, f(y_i)$ is generated by a sub-circuit of C_N which takes as inputs $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$ and $\mathsf{tt}(f)$, such that the only gates in this sub-circuit with fan-out > 1 are the inputs $y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1})$.

Our main contribution is the second part of Lemma 4.39, but we also give a more self-contained proof. We use linear hash functions along with the assumption that $\mathsf{NP} \subseteq \mathsf{NC}^1$, to make the counting arguments in the proof constructive. Additionally, we use the *Valiant-Vazirani isolation lemma* in the process of selecting the anti-checker instances, to ensure that the anti-checker construction has an almost-formula structure.

For the proof of Lemma 4.39, we need the following results. Firstly, we state the classical Valiant-Vazirani isolation lemma.

Lemma 4.40 (Valiant-Vazirani isolation lemma (Lemma 17.19 in [AB09]))**.** *Let $\mathcal{H}_{n,k}$ be a pairwise independent hash function collection*[14] *from $\{0,1\}^n$ to $\{0,1\}^k$, where $2 \leq k \leq$*

[14] A collection of functions $\mathcal{H}_{n,k}$ from $\{0,1\}^n$ to $\{0,1\}^k$ is called pairwise independent, if for every $x, x' \in \{0,1\}^n$ with $x \neq x'$, and for every $y, y' \in \{0,1\}^k$, $\Pr_{h\sim\mathcal{H}_{n,k}}\{(h(x) = y) \wedge (h(x') = y')\} = 1/2^{2k}$.

$n+1$, *and* $S \subseteq \{0,1\}^n$ *such that* $2^{k-2} \leq |S| \leq 2^{k-1}$. *Then*

$$\Pr_{h \sim \mathcal{H}_{n,k}} \{\textit{there exists a unique } x \in S \textit{ such that } h(x) = 0^k\} \geq 1/8$$

We also need the following result for approximate counting using linear hash functions.

Lemma 4.41 ([Jer09] (Paragraph 2 in Section 3))**.** *For any* $s \leq 2^n$ *and* $\varepsilon > 0$, *let* $c = 2(1/\varepsilon \cdot (\log\log s + \log(1/\varepsilon)))$, *and* $t_M = \log(4s^c)$. *If* $X \subseteq \{0,1\}^n$ *such that* $|X| = s$, *then there exists matrices* $A_1, \ldots, A_{t_M} \in \{0,1\}^{t_M \times cn}$ *such that*

- *Each* $A_i : \{0,1\}^{cn} \to \{0,1\}^{t_M}$ *defines a linear mapping from* X^c *(the c times Cartesian product of* X*) to* $\{0,1\}^{t_M}$.
- *For each* A_i, *where* $1 \leq i \leq t_M$, *there exists a set* $X_i \subseteq X^c$ *which is mapped injectively by* A_i, *i.e. for every* $x \in X_i$ *and for every* $y \in X^c$, *if* $y \neq x$ *then* $A_i(y) \neq A_i(x)$. *Moreover,* $\cup_i X_i = X^c$.

In particular, if $|X| \leq s$, *then there exists an injective mapping (provided by the* A_i*s) from* X^c *to a set of size at most* $t_M \cdot 2^{t_M} \leq s^c(1+\varepsilon)^c$.

The first part of the theorem is proved using the probabilistic method. Indeed, the matrices $A_1, \ldots, A_{t_M}$ are picked uniformly at random and over the random choices of these matrices, observe that the probability that there exists an element in X^c which is not mapped injectively by any A_i is very low. In other words, there exists a choice of matrices, such that for each element in X^c, there exists an A_i which injectively maps it into $\{0,1\}^{t_M}$.

For the second part, consider the following mapping $\mathcal{M}$ from X^c to a set $\mathcal{A}$ of size $t_M 2^{t_M}$, comprised of t_M blocks of size 2^{t_M} each: $\mathcal{M}$ maps each element $z \in X^c$ into the i^{th} block in $\mathcal{A}$, if A_i is the first matrix which maps z injectively (i.e. $z \in X_i$). Since each $z \in X^c$ is mapped injectively by some A_i, we see that $\mathcal{M}$ injects X^c into a set of size at most $t_M 2^{t_M} \leq 4s^c \log(4s^c) \leq s^c(1+\varepsilon)^c$, for the values of c and t_M chosen.

We build some notation before diving into the proof. For any function f on n inputs, $y_1, \ldots, y_i \in \{0,1\}^n$ and $\lambda > 0$, define $S_f(y_1, \ldots, y_i)$ as the set of circuits of size at most $2^{n^\lambda}/2n$ which are consistent with $y_1, \ldots, y_i$ on f, $R_f(y_1, \ldots, y_i) = |S_f(y_1, \ldots, y_i)|$, and $\tau_f(y_1, \ldots, y_i)$ as the fraction of these circuits with respect to all the circuits of size $2^{n^\lambda}/2n$.[15] Note that, any circuit C is consistent with a set of inputs S on f, if $C(z) = f(z)$ for every $z \in S$. We also need the following combinatorial claim from [OPS19].

[15] Of course, S_f, R_f and τ_f are also functions of the values of f on the y_i's. We omit it in the notation for simplicity.

Proposition 4.42 (Lemma 23 in [OPS19]). *For every $i \geq 1$, $\lambda > 0$, any function $f \notin \mathsf{SIZE}[2^{n^\lambda}]$ and every $y_1, \ldots, y_{i-1} \in \{0,1\}^n$, suppose that $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$ and $\tau_f(y_1, \ldots, y_{i-1}) \leq (1-1/4n)^{i-1}$. Then, there exists $y_i \in \{0,1\}^n$ such that $\tau_f(y_1, \ldots, y_i) \leq (1-1/4n)^{i-1}(1-1/2n)$.*

Proof of Lemma 4.39. Suppose that $\mathsf{NP} \subseteq \mathsf{NC}^1$. For any $\lambda \in (0,1)$, let f be any function on n variables such that $f \notin \mathsf{SIZE}[2^{n^\lambda}/2n]$. Let $w = \mathsf{tt}(f) \in \{0,1\}^N$ be the input to the circuit C_N which we'll construct. The construction of the circuit is iterative, where at the i^{th}-stage, $y_i \in \{0,1\}^n$ is added to the current anti-checker set $\{y_1, \ldots, y_{i-1}\}$, by maintaining the invariant that $\tau_f(y_1, \ldots, y_i) \leq (1-1/4n)^i$ (if $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$). C_N outputs $y_1, \ldots, y_t$ along with $f(y_1), \ldots, f(y_t)$, for $t = 2^{O(n^\lambda)}$ (the reason for this choice of t will be seen later in the proof).

Let $t' = t - 2n^2$. Assume that for some stage $i < t'$, $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$, and at the beginning of the i^{th}-stage, we have generated $y_1, \ldots, y_{i-1} \in \{0,1\}^n$ as the current instances in the anti-checker, such that $\tau_f(y_1, \ldots, y_{i-1}) \leq (1-1/4n)^{i-1}$ (when $i = 1$, no strings have been generated and trivially, all circuits are consistent with the anti-checkers generated so far). Our goal is to find a string $y_i \in \{0,1\}^n$ such that $\tau_f(y_1, \ldots, y_{i-1}, y_i) \leq (1-1/4n)^i$. We call any such string y_i as "good".

For any candidate $y^* \in \{0,1\}^n$, we first show the existence of a formula F that takes an input $w = (y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1}), y^*, f(y^*))$ of size $O(tn)$ (the input is suitably padded to achieve this length, if necessary), and checks if $\tau_f(y_1, \ldots, y_{i-1}, y^*)$ has shrunk by a non-trivial factor. More formally,

$$\begin{aligned} &\tau_f(y_1, \ldots, y_{i-1}, y^*) \leq (1-1/4n)^{i-1}(1-1/2n) \implies F(w) = 1 \\ &F(w) = 1 \implies \tau_f(y_1, \ldots, y_{i-1}, y^*) \leq (1-1/4n)^i \end{aligned} \tag{4.1}$$

To construct the formula, we use the approximate counting method from Lemma 4.41, by setting X as $S_f(y_1, \ldots, y_{i-1}, y^*)$, $\varepsilon = 1/4n$, $t_M = 2^{O(n^\lambda)}$, and $c = \mathsf{poly}(n)$ (for a large enough polynomial). If y^* is such that $\tau_f(y_1, \ldots, y_{i-1}, y^*) \leq (1-1/4n)^{i-1}(1-1/2n)$, then from Lemma 4.41, there exists matrices $A_1, \ldots, A_{2^{O(n^\lambda)}}$ such that X^c is injectively mapped into the Cartesian power (with exponent c) of a set of size $(1-1/4n)^{i-1}(1-1/2n)(1+4n) \cdot 2^{2^{O(n^\lambda)}} \leq (1-1/4n)^i \cdot 2^{2^{O(n^\lambda)}}$. On the other hand, the existence of such matrices witness $\tau_f(y_1, \ldots, y^*) \leq (1-1/4n)^i$ (by definition). Furthermore, on input w, the existence of such matrices is a Σ_p^2-property.[16] Since $\mathsf{NP} \subseteq \mathsf{NC}^1$, we get a formula F of size $\mathsf{poly}(n,t) = 2^{O(n^\lambda)}$ which decides the existence of these matrices and thus, satisfies the properties in Equation 4.1.

[16] Accept input $w = (y_1, \ldots, y_{i-1}, y^*, f(y_1), \ldots, f(y_{i-1}), f(y^*))$ (padded to length $O(tn) = 2^{O(n^\lambda)}$), if $\exists$ matrices $A_1, \ldots, A_t \in \{0,1\}^{t \times cn}$, $\forall x \in X^c, \forall y_1, \ldots, y_t \in X^c$, $\bigvee_{i=1}^t (x \neq y_i \implies A_i(x) \neq A_i(y_i))$. This is a Σ_p^2 property.

From Proposition 4.42, observe that for $w_{i-1} = (y_1, \ldots, y_{i-1}, f(y_1), \ldots, f(y_{i-1}))$, there exists $y^* \in \{0,1\}^n$ such that $\tau_f(y_1, \ldots, y_{i-1}, y^*) \leq (1 - 1/4n)^{i-1}(1 - 1/2n)$, i.e. there exists y^* such that $F(w_{i-1}, y^*, f(y^*)) = 1$. To pick a good y_i, we use F in an exhaustive search over all n-bit strings, along with Lemma 4.40 to isolate a good string and make its selection simpler. In more detail, for any $z \in \{0,1\}^n$, $k \leq n+1$ and $h \in \mathcal{H}_{n,k}$, where $\mathcal{H}_{n,k}$ is an efficiently computable, pairwise independent hash function family from $\{0,1\}^n$ to $\{0,1\}^k$, define

$$F^{k,h}(w_{i-1}, z, f(z)) = F(w_{i-1}, z, f(z)) \wedge (h(z) = 0^k) \tag{4.2}$$

The input to $F^{k,h}$ is of size $O(tn)$. On input z, $h(z) = 0^k$ can be checked using a $\mathsf{poly}(n)$-sized formula (again, since $\mathsf{NP} \subseteq \mathsf{NC}^1$). Thus, $F^{k,h}$ is a formula of size $2^{O(n^\lambda)}$.

From Lemma 4.40, we see that for any fixed w_{i-1}, if h is picked uniformly at random from $\mathcal{H}_{n,k}$ and k is also chosen randomly from $\{2, \ldots, n+1\}$, then with probability at least $1/8n$, there is a unique z satisfying Equation 4.2. Randomly picking $2^{d \cdot n^\lambda}$ many tuples (k,h) (where the constant d in the exponent is large enough), implies that the probability that $F^{k,h}$ does not have a unique solution for any of them is at most $(1 - 1/8n)^{2^{d \cdot n^\lambda}} < 2^{-\frac{2^{d \cdot n^\lambda}}{8n}}$. Thus, we can non-uniformly fix a set $\mathcal{R}$ of $2^{O(n^\lambda)}$ tuples such that for each w_{i-1}, at least one tuple (k,h) from $\mathcal{R}$ will ensure that $F^{k,h}(w_{i-1}, z, f(z))$ has a unique satisfying assignment.

Consequently, for every (k,h) in $\mathcal{R}$, and every $j \in \{0,1\}^{\log n}$, we define the following formula $D_j^{k,h}$ on input w_{i-1}, which outputs the j^{th}-bit of a candidate element for the anti-checker.

$$D_j^{k,h}(w_{i-1}) = \bigvee_{b \in \{0,1\}^n} (b_j \wedge F^{k,h}(w_{i-1}, b, f(b))) \tag{4.3}$$

Note that $D_j^{k,h}$ is $2^{n+O(n^\lambda)}$-sized formula. Going over all tuples in $\mathcal{R}$, we get $2^{O(n^\lambda)}$ candidate strings, one of which is assured to be good as there exists at least one tuple $(k,h) \in \mathcal{R}$ such that $F^{k,h}$ has a unique solution. Since $\mathsf{NP} \subseteq \mathsf{NC}^1$, given the list of $2^{O(n^\lambda)}$ candidate strings, selecting a good y_i from this list can be done by a formula of size $2^{O(n^\lambda)}$. Having y_i, a formula of size $N \log N$ with access to $\mathsf{tt}(f)$ outputs $f(y_i)$. Overall, $y_i, f(y_i)$ can be generated by a circuit of size at most $2^{n+O(n^\lambda)}$, using w_{i-1} as input.

The iterative process for constructing an anti-checker is as follows. For any $i < t'$, we check if $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$. Given input w_{i-1}, one can check if $R_f(y_1, \ldots, y_{i-1}) \geq 2n^2$, using an NP-property. Since $\mathsf{NP} \subseteq \mathsf{NC}^1$, this can be done using a formula of $2^{O(n^\lambda)}$. If this is true, then a circuit of size $2^{n+O(n^\lambda)}$ generates $y_i, f(y_i)$ such that $\tau_f(y_1, \ldots, y_i) \leq (1 - 1/4n)^i$.

Since $(1-1/4n)^{2^{O(n^\lambda)}} < 1/2^{2^{O(n^\lambda)}/4n}$, the number of steps t' after which $R_f(y_1, \ldots, y_i) < 2n^2$ is at most $2^{O(n^\lambda)}$. At this stage, the remaining circuits can be generated by a Σ_p^2

algorithm as a string σ (which encodes at most $2n^2$ many circuits) (see Lemma 21 of [OPS19]) on input $w_{t'}$. Since $\mathsf{NP} \subseteq \mathsf{NC}^1$, σ can be generated by a formula of size at most $2^{O(n^\lambda)}$. For each of these circuits, we use a similar strategy as earlier, of using the Valiant-Vazirani isolation lemma over each element in $\mathcal{R}$ and generate an instance on which the circuit does not agree with f. The only difference is that instead of the formula $F^{k,h}$, we use a new formula $E^{k,h}$ for each $(k,h) \in \mathcal{R}$, which takes an encoding of a circuit A of size $2^{n^\lambda}/2n$, an input $x \in \{0,1\}^n$ to A and $f(x)$ as inputs, and accepts iff $(A(x) \neq f(x)) \wedge (h(x) = 0^k)$.[17] It is worth noting that we generate σ using input $w^{t'}$ each time $E^{k,h}$ is computed, which is still at most $2^{n+O(n^\lambda)}$ many times. Thus, for each remaining circuit A in σ, we get a formula of size $2^{n+O(n^\lambda)}$ which outputs an anti-checker element on which A does not agree with f, using $\mathsf{NP} \subseteq \mathsf{NC}^1$. The number of anti-checker elements generated is at most $t = t' + 2n^2 = 2^{O(n^\lambda)}$.

Put together, the anti-checkers are generated by a circuit C_N of size at most $2^{n+O(n^\lambda)}$. Furthermore, the circuit has a $2^{O(n^\lambda)}$-almost-formula structure, since all the intermediate computations are done by formulas (many of which crucially use $\mathsf{NP} \subseteq \mathsf{NC}^1$), and the only gates with fan-out larger than 1 are those which generate y_i and $f(y_i)$. □

We now prove the hardness magnification result.

Proof of Theorem 4.38. The proof follows from a similar approach in [OPS19] (see Theorem 4). We prove that if $\mathsf{NP} \subseteq \mathsf{NC}^1$, $\mathsf{MCSP}[2^{\sqrt{n}}/2n, 2^{\sqrt{n}}] \in 2^{O(\sqrt{n})}\text{-}\mathsf{Almost\text{-}Formula}[N^{1+\varepsilon}]$ for every $\varepsilon > 0$. The idea is that on input $\mathsf{tt}(f) \in \{0,1\}^N$, we first use $\mathsf{NP} \subseteq \mathsf{NC}^1$ in Lemma 4.39, to get an anti-checker $\{y_1, \ldots, y_t\}$ and its evaluations $f(y_1), \ldots, f(y_t)$ for $t = 2^{O(\sqrt{n})}$, using an $2^{O(\sqrt{n})}$-almost-formula of size $2^{n+O(\sqrt{n})}$.

We then reduce gap MCSP to the following problem L (also called as Succinct-MCSP): On input $(1^n, 1^s, 1^t, (x_1, b_1), \ldots, (x_t, b_t))$, where each $x_i \in \{0,1\}^n$ and $b_i \in \{0,1\}$, accept if and only if it is of the right format and there exists a circuit E over n variables of size at most s, such that $E(x_i) = b_i$ for every $1 \leq i \leq t$. Note that this problem is in NP, and from our assumption can be computed by formulas of polynomial size.

The reduction passes the anti-checker $\{y_1, \ldots, y_t\}$ along with $f(y_1), \ldots, f(y_t)$ to L, with $s = 2^{\sqrt{n}}/2n$ and $t = 2^{O(\sqrt{n})}$, and outputs its answer. If f can be computed by a circuit of size s, then for any choice of $y_1, \ldots, y_t$, L accepts it. On the other hand, if L cannot be computed by circuits of size $2^{\sqrt{n}}$, every circuit of size s fails on at least one of the y_i's and thus, L always rejects its input.

Furthermore, since the input length to L is $O(tn)$, there exists a formula of size $\mathsf{poly}(t,n) = 2^{O(\sqrt{n})}$ which decides L. Composing this formula into the reduction, we get

[17] Such a formula exists since the circuit value problem, which takes as inputs $x \in \{0,1\}^n$ and a suitable encoding of an n-variate circuit A of size s, and outputs $A(x)$, is P-Complete, and $\mathsf{NP} \subseteq \mathsf{NC}^1$.

an $2^{O(\sqrt{n})}$-almost-formula of size $2^{n+O(\sqrt{n})} \leq N^{1+\varepsilon}$ which computes $\mathsf{MCSP}[2^{\sqrt{n}}/2n, 2^{\sqrt{n}}]$, for every $\varepsilon > 0$. Note that the construction from Lemma 4.39 ensures that the only gates with fan-out > 1 are those which compute $y_i, f(y_i)$, for each $i \leq t$. □

Items 2 [RR97] and 4 [HS17, OPS19] follow from known results. Next, we show Item 3, a lower bound for Parity against almost-formulas.

Let F be a t-almost-formula. Moving bottom-up, observe that each gate G which has fan-out > 1 is computed by a formula whose inputs are either the original inputs or other gates with fan-out > 1. We call such a maximal formula as a *principal formula* at G.

Lemma 4.43. *For any* $0 < \varepsilon < 1/6$, $\mathsf{Parity} \notin n^{\varepsilon}\text{-}\mathsf{Almost\text{-}Formula}[n^{2-6\varepsilon}]$

Proof. For some small enough $\varepsilon > 0$, let $t = n^{\varepsilon}$. Suppose that Parity can be computed by t-almost-formulas $\{G_n\}$ of size at most $n^{2-6\varepsilon}$. We show that this implies the existence of a sequence of formulas $\{F_n\}$ of size $n^{2-5\varepsilon}$ which compute Parity on at least $(1/2 + 1/2^{n^{\varepsilon}+1})$-fraction of the inputs.

Observe that G_n has at most t gates with fan-out > 1, i.e. it has at most t principal formulas $F_1, \ldots, F_t$ (assuming some pre-defined ordering). Thus, there exists a constant vector $a \in \{0,1\}^t$, such that for at least $2^n/2^t$ many inputs $y \in \{0,1\}^n$, $F_i(y) = a_i$ for every $1 \leq i \leq t$. Further, let $b \in \{0,1\}$ be the majority output of G_n over the rest of the inputs (inputs y such that there exists some F_i for which $F_i(y) \neq a_i$).

The formula F_n gets a and b as non-uniform advice, and on input $x \in \{0,1\}^n$, checks if each principal formula $F_i(x) = a_i$. If this is the case, then F fixes the gates in the original almost-formula with its corresponding constants and computes the output using the resulting formula, otherwise it outputs b. Clearly, the size of F_n is at most $n^{2-5\varepsilon}$, and it approximates Parity on at least $(1/2 + 1/2^{n^{\varepsilon}+1})$-fraction of the inputs.

On the other hand, recall from Lemma 4.32 that any function f on n inputs which can be computed by a formula of size s, is pointwise approximated by a real, multilinear polynomial of degree $O(\sqrt{s})$ up to an additive error of $1/3$. Further, this can be generalised to saying that f can be approximated up to a pointwise additive error of $1/2^r$, by a real polynomial of degree $O(r\sqrt{s})$ (cf. Fact 5.9 in [Tal17a]). However, [KR13] (Section 1.2) state that this implies that any formula of size $o\left((\frac{n}{r})^2\right)$ computes Parity on at most $(1/2 + 1/2^{r+O(1)})$-fraction of the n-bit inputs. Taking $r = n^{2\varepsilon}$, we see that any formula of size $o(n^{2-4\varepsilon})$ computes Parity on at most $(1/2 + 1/2^{n^{2\varepsilon}+O(1)})$-fraction of the inputs, leading at a contradiction. □

4.5.2 Locality Barrier for Almost-Formula Lower Bounds Above the Threshold

This section captures an instantiation of the locality barrier for HM Frontier C. Recall the definition of principal formulas of an almost-formula from the last section. Let F be a t-almost-formula. Each gate G which has fan-out > 1 is computed by a formula whose inputs are either the original input variables or other gates with fan-out > 1. Any such a maximal formula is called as a *principal formula* at G.

Theorem 4.44 (Locality Barrier for HM Frontier C)**.** *The following results hold.*

(C1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) $\mathsf{MCSP}[2^{\sqrt{n}}/2n, 2^{\sqrt{n}}]$ *is computable by oracle* $2^{O(\sqrt{n})}$*-almost-formulas of size* $2^{n+O(\sqrt{n})}$ *with oracle gates of fan-in* $2^{O(\sqrt{n})}$*, such that the oracles are only at the bottom layer of any principal formulas.*

(C3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques) *For every* $0 < \varepsilon < 1/8$, $\mathsf{Parity} \notin n^{\varepsilon}$-$\mathsf{Almost\text{-}Formula}[n^{2-8\varepsilon}]$*, even if the almost-formulas are allowed to use arbitrary oracle gates of fan-in* $< n^{\varepsilon}$ *which are present only at the bottom layer of any principal formulas.*

Proof. The first item follows by inspecting the proof of Theorem 4.38. We see that $\mathsf{MCSP}[2^{\sqrt{n}}/2n, 2^{\sqrt{n}}]$ is computable by $2^{O(\sqrt{n})}$-almost-formulas $\{F_N\}$ of size $2^{n+O(\sqrt{n})}$ with local oracles of fan-in $2^{O(\sqrt{n})}$. Moreover, the only gates of fan-out larger than 1 are the gates outputting the anti-checker elements $y_1, \ldots, y_{2^{O(\sqrt{n})}}$ with bits $f(y_1), \ldots, f(y_{2^{O(\sqrt{n})}})$. We want to show that the local oracle gates are only at the bottom of principal formulas. In order to achieve this we need to modify formulas F_N slightly.

First, note that F_N has an oracle gate which is applied on top of the final anti-checker $y_1, \ldots, y_{2^{O(\sqrt{n})}}$ with bits $f(y_1), \ldots, f(y_{2^{O(\sqrt{n})}})$. In order to ensure that this oracle is at the bottom of a principal formula that computes a gate with fan-out > 1, we simply add dummy negation gates to the output gate, and the gates computing the anti-checker $y_1, \ldots, y_{2^{O(\sqrt{n})}}$ with bits $f(y_1), \ldots, f(y_{2^{O(\sqrt{n})}})$, if necessary.

Second, note that each anti-checker element y_{i+1}, along with $f(y_{i+1})$ is generated as follows: (1) If $R_f(y_1, \ldots, y_i) \geq 2n^2$, then a subformula F_1 generates $(y_{i+1}, f(y_{i+1}))$ (2) or else, $R_f(y_1, \ldots, y_i) < 2n^2$, and a subformula F_2 generates at most $2n^2$ anti-checker elements. In both cases, the predicates $R_f(y_1, \ldots, y_i) < 2n^2$ are checked by an oracle.

In case 1, for every $(k, h) \in \mathcal{R}$ and $1 \leq j \leq n$, $D_j^{k,h}$ is a subformula of F_1 with oracles at the bottom, as seen in the proof of Lemma 4.39. This process generates a set of $2^{O(\sqrt{n})}$ candidate anti-checker elements. F_1 chooses a good string by applying another oracle. In order to ensure that the latter oracle gate is at the bottom of a principal formula, we add dummy negation gates to the gates generating the potential good strings. This increases the number of gates with fan-out larger than 1 only by $2^{O(\sqrt{n})}$.

In case 2, a similar analysis as case 1 works, where for each subformula which generates candidate anti-checker elements for each remaining circuit, we ensure that the oracles are at the bottom by asking them to perform both tasks: choose the next remaining circuit A and compute $(A(x) \neq f(x)) \wedge (h(x) = 0^k)$ on a given input x (and fixed $(k, h) \in \mathcal{R}$). The oracle selecting the right anti-checker element from the set of potential instances is also treated in the same way as in case 1. All in all, we obtain an almost-formula with the desired structure.

The second item is proved in analogous fashion to Lemma 4.43. Suppose that Parity has n^ε-almost-formulas of size $n^{2-8\varepsilon}$, with local oracles at the bottom of principal formulas. Since there are only n^ε gates of fan-out > 1, we can replace these gates by constants and use ideas seen earlier to obtain formulas F_n of size $n^{2-7\varepsilon}$ with local oracles at the bottom computing Parity with probability $\geq 1/2 + 1/2^{n^\varepsilon+1}$.

Let $L'(f)$ be the size (i.e. the number of leaves) of the smallest formula with local oracles at the bottom computing f. Now any function $f : \{-1, 1\}^n \to \{-1, 1\}$ computed by an oracle formula, where any oracle gate has fan-in at most n^ε and is located at the bottom, can be approximated by a polynomial of degree $O(tn^\varepsilon\sqrt{L'(f)})$ up to pointwise error of 2^{-r} (cf. Lemma 4.33 for some intuition about this). This implies that any formula of size $o((n/r)^2(1/n^\varepsilon)^2)$ with local oracles at the bottom computes Parity with probability at most $1/2 + 1/2^{r+O(1)}$ (for large enough t). Taking $t = n^{2\varepsilon}$, we see that any such oracle formula of size $o(n^{2-6\varepsilon})$ computes Parity on at most $(1/2 + 1/2^{O(n^{2\varepsilon})})$-fraction of the inputs, leading to a contradiction. □

4.6 HM Frontier for Gap MCSP and One-sided Error Randomised Formulas

A one-sided error randomised formula of size s is a distribution $\mathcal{F}$ over De Morgan formulas of size s [CJW20]. Any Boolean function f over n inputs is said to be computed by $\mathcal{F}$, if for every $x \in \{0, 1\}^n$

- if $f(x) = 1$, then $\Pr_{F\sim\mathcal{F}}\{F(x) = 1\} = 1$.
- if $f(x) = 0$, then $\Pr_{F\sim\mathcal{F}}\{F(x) = 0\} \geq 0.9$.

Define GapAND_N as the promise problem on N bits such that it outputs 1 when all input bits are 1, and outputs 0 when at most $1/10$ of the input bits are 1. W.l.o.g. we assume that a one-sided error randomised formula is equivalent to a $\mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[s]$ circuit. In one direction, given such a circuit, the uniform distribution over the formulas which are inputs to the $\mathsf{GapAND}_{O(N)}$ gate gives a one-sided randomised

formula over $\mathsf{Formula}[s]$. Indeed, for any input x, if $f(x) = 1$, then the sampled formula always outputs 1, whereas it outputs 0 with probability at least 0.9, if $f(x) = 0$.

In the other direction, given a randomised formula $\mathcal{F}$ of size s (although with a slightly less error of at most 0.09 when $f(x) = 0$), one can sample $O(N)$ many formulas and use an argument akin to Adleman's trick for non-uniform derandomisation, to obtain a $\mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[s]$ circuit.

We first establish HM Frontier D in Section 4.6.1. Following this, we show that known lower bound techniques by [Hås98] and [CKLM19] localise in Section 4.6.2.

4.6.1 Establishing HM Frontier D

We first prove the following hardness magnification theorem.

Theorem 4.45 (HM for worst-case MCSP via kernelisation)**.** *If there exists $\varepsilon > 0$, such that $\mathsf{MCSP}[2^{n^{1/3}}] \notin \mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}$-$\mathsf{XOR}[N^{\varepsilon}]$, then $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$.*

The proof follows by an adaptation of the kernelisation-based approach from [CJW19].

Proof of Theorem 4.45. We prove the contrapositive of the statement. Define $s(n) = 2^{n^{1/3}}$ and $\mathcal{S}$ as the set of truth tables which can be computed by circuits of size at most $s(n)$. Let $E_N : \{0,1\}^N \to \{0,1\}^M$ be the error-correcting code from Theorem 4.29, where $M = M(N) = b \cdot N$ for some constant $b > 0$.

For a positive integer T to be fixed later, we define a family of functions $\{H_v : \{0,1\}^N \to \{0,1\}^T\}$, where each function is specified by a seed v of length $T \log M$. For any v and input $x \in \{0,1\}^N$, H_v computes $E_N(x)$, splits v into $(v_1, \ldots, v_T)$ where each $v_i \in [M]$ and outputs $h = (E_N(x)_{v_1}, \ldots, E_N(x)_{v_T})$. Informally, $H_v(x)$ computes the codeword $E_N(x)$ and outputs a T-length random projection of $E_N(x)$, specified by the indices in the uniformly random seed v.

Firstly, observe that with high probability over the random choice of the seed v, every input $x \in \mathcal{S}$ is mapped to a distinct string in $\{0,1\}^T$ by H_v.[18] Indeed, for any pair of inputs $x, y \in \mathcal{S}$ where $x \neq y$, since the minimum distance for the code generated by E_N is greater than 2δ, for the constant $\delta > 0$ given by Theorem 4.29, $\Pr_{i \in [M]}\{E_N(x)_i = E_N(y)_i\} \leq (1 - 2\delta)$. Thus, $\Pr_v\{H_v(x) = H_v(y)\} \leq \Pr_v\{\forall i \in [T], E_N(x)_{v_i} = E_N(y)_{v_i}\} \leq (1 - 2\delta)^T \leq e^{-2T\delta}$. Setting $T = c \cdot s(n) \log s(n)$ for a large enough constant $c > 0$ and using a union bound over all pairs of strings in $\mathcal{S}$ (of size $\binom{|\mathcal{S}|}{2}$), with probability at least 0.99 over the choice of v, for each pair of distinct $x, y \in \mathcal{S}$, $H_v(x) \neq H_v(y)$. Let v^* be such a good seed.

[18] One can consider $\{H_v\}$ as a k-perfect linear hash function family of seed length $T \log M$, i.e. for every subset S of inputs of size at most $k = 2^{s(n) \log s(n)}$, there exists a seed v such that H_v is injective over S.

Next, define the language $L : [M]^T \times \{0,1\}^T \times [M] \times \{0,1\} \to \{0,1\}$ as follows. An input $w \in L$, if w is of the form (v, h, i, a) and there exists $y \in \{0,1\}^N$ such that $y \in \mathcal{S}$, $H_v(y) = h$ and $E_N(y)_i = a$. Since, $T = O(s(n) \cdot \mathsf{poly}(n)) = 2^{O(n^{1/3})}$, we see that $L \in \mathsf{NQP}$.

Suppose that $\mathsf{NQP} \subseteq \mathsf{Formula}[n^k]$, for some constant $k > 0$. For any $\varepsilon > 0$, we construct $M = O(N)$ $\mathsf{Formula\text{-}XOR}$ circuits $C_1, \ldots, C_M$ of size N^ε each, such that for any $x \in \mathcal{S}$, $C_i(x) = 1$ for all $i \in [M]$, and otherwise $C_i(x) = 0$ for at least a constant fraction of the i's. Indeed, for any $i \in [M]$, $C_i(x)$ gets v^* as non-uniform advice, computes $H_{v^*}(x)$ using T XOR gates and then outputs $L(v^*, H_{v^*}(x), i, E_N(x)_i)$. The input length to L is $O(T \log N) = N^{o(1)}$ and since L can be computed by formulas of size $O(T^k)$, each C_i is a $\mathsf{Formula\text{-}XOR}$ circuit of size N^ε.

For the correctness of the reduction, if $x \in \mathcal{S}$, L accepts $(v^*, H_{v^*}(x), i, E_N(x)_i)$ for every $i \in [M]$, by just making the non-deterministic guess as x. On the other hand, if $x \notin \mathcal{S}$, the choice of v^* ensures that there exists at most one $y \in \mathcal{S}$ such that $H_{v^*}(x) = H_{v^*}(y)$. For the case that there exists no such y, L rejects $(v^*, H_{v^*}(x), i, E_N(x)_i)$ for every i. Otherwise, since $x \neq y$, there exists some $j \in [M]$ such that $E_N(x)_j \neq E_N(y)_j$ and L rejects the input $(v^*, H_{v^*}(x), j, E_N(x)_j)$. In particular, it is worth noting that if $x \notin \mathcal{S}$, L rejects $(v^*, H_{v^*}(x), i, E_N(x)_i)$ for at least a constant fraction (a 2δ-fraction) of the i's.

Finally, we construct M many $\mathsf{Formula\text{-}XOR}$ circuits $D_1, \ldots, D_M$, where each D_i runs C_i over constantly many independently sampled seeds and reduces the error to at most 0.09 when $x \notin \mathcal{S}$ (we fix constantly many good seeds in the non-uniform advice v^*). Thus, we have M many $\mathsf{Formula\text{-}XOR}$ circuits $D_1, \ldots, D_M$ of size $O(N^\varepsilon)$ each, such that if $x \in \mathcal{S}$, then $D_i(x) = 1$ for all $i \in [M]$, and else $D_i(x) = 0$ for at least 0.91 fraction of the i's, which is nothing but a one-sided error randomised $\mathsf{Formula\text{-}XOR}[N^\varepsilon]$ circuit. This in turn means that we get a $\mathsf{GapAND}_{O(N)}\text{-}\mathsf{Formula\text{-}XOR}[N^\varepsilon]$ circuit computing $\mathsf{MCSP}[s]$. □

Since XOR gates can be computed by formulas of quadratic size, Theorem 4.45 immediately proves item D1.

Corollary 4.46. *If there exists $\varepsilon > 0$, where $\mathsf{MCSP}[2^{n^{1/3}}] \notin \mathsf{GapAND}_{O(N)}\text{-}\mathsf{Formula}[N^{2+\varepsilon}]$, then $\mathsf{NQP} \not\subseteq \mathsf{NC}^1$.*

Theorem 4.45 also gives an alternative way of proving Item 1 in HM Frontier B, i.e. Theorem 4.28. This can be achieved by replacing the GapAND-gate with the standard AND gate, for derandomising the randomised formula in the proof.

Regarding the other items in HM Frontier D, items D2 ([RR97]) and D4 ([CJW20]) follow from known results. In particular, item D4 is proved by techniques which build on similarly proved lower bounds in [HS17, OPS19], via iterated pseudo-random restrictions. Finally, both lower bounds in item D3 which are implicit in previous works [Hås98, CKLM19] are also implied by stronger results proved in Section 4.6.2.

4.6.2 Locality Barrier for Randomised Formula Lower Bounds Above the Threshold

In this section we capture an instantiation of the locality barrier for HM Frontier D.

Theorem 4.47 (Locality Barrier for HM Frontier D)**.** *The following hold:*

(D1$^{\mathcal{O}}$) *(Oracle Circuit Construction from Magnification) There exists $\varepsilon, \beta > 0$ and an oracle $\mathcal{O}$, such that* $\mathsf{MCSP}[2^{n^{1/3}}] \in \mathsf{GapAND}_{O(N)}\text{-}\mathcal{O}_{N^\beta}\text{-}\mathsf{Formula}[N^{2+\varepsilon}]$.

(D3$^{\mathcal{O}}$) *(Extension of Lower Bound Techniques - 1) For every $\beta > 0$, every $1 > \varepsilon > \beta$ and any oracle $\mathcal{O}$,* $\mathsf{Andreev}_{2N} \notin \mathsf{GapAND}_{O(N)}\text{-}\mathcal{O}_{N^\beta}\text{-}\mathsf{Formula}[N^{3-\varepsilon}]$.

(D3$^{\mathcal{O}}$) *(Extension of Lower Bound Techniques - 2) For every $\beta > 0$, every $1 > \varepsilon > 3\beta$ and any oracle $\mathcal{O}$,* $\mathsf{MCSP}[2^n/\mathsf{poly}(n)] \notin \mathsf{GapAND}_{O(N)}\text{-}\mathcal{O}_{N^\beta}\text{-}\mathsf{Formula}[N^{3-\varepsilon}]$.

Proof of Item 1 of Theorem 4.47. The first item can be observed directly from the construction given in Corollary 4.46. □

Define Andreev's function as $\mathsf{Andreev}_{2N} : \{0,1\}^{2N} \to \{0,1\}$, where $\mathsf{Andreev}$ takes two inputs $x, y \in \{0,1\}^N$ and does the following. It partitions x into $\log N$ blocks $x_1, \ldots, x_{\log N}$ of size $r = N/\log N$ each and computes $j = (\mathsf{XOR}(x_1), \ldots, \mathsf{XOR}(x_{\log N}))$. It then treats y as a truth table of a function $f_y : \{0,1\}^{\log N} \to \{0,1\}$ and outputs y_j.

Classical results by [Hås98, Tal14] show that $\mathsf{Andreev}_{2N}$ cannot be computed by sub-cubic formulas, using the fact that any small-sized formula shrinks significantly under a suitably chosen random restriction (Theorem 4.10). In Item 2 of Theorem 4.47, we show that this lower bound technique localises, even for the case of GapAND-Formulas.

Proof of Item 2 Theorem 4.47. For some $\beta > 0$, suppose $\mathsf{Andreev}_{2N}$ can be computed by a $\mathsf{GapAND}_{O(N)}\text{-}\mathcal{O}_{N^\beta}\text{-}\mathsf{Formula}$ formula of size at most $N^{3-\varepsilon}$. Apply a p-regular truly random restriction ρ sampled from $\mathcal{R}_p$ over the variables in x, for $p = O(\log N \log\log N)/N$.

Fix the inputs in y to some string in $\{0,1\}^N$. Observe that with probability at least 0.95, each block x_i has a variable which is unrestricted by ρ. Next, under the application of ρ, we use Theorem 4.10 to see that every $\mathcal{O}_{N^\beta}\text{-}\mathsf{Formula}$ F of size $N^{3-\varepsilon}$ collapses to an $\mathcal{O}_{N^\beta}\text{-}\mathsf{Formula}$ of size at most $N^{1-\varepsilon} \cdot \mathsf{poly}(\log N)$ in expectation. In particular, every oracle formula F_i which is an input to the GapAND gate collapses in expectation. Using Markov's inequality (Lemma 2.5), we see that for each oracle formula F_i, there exists a fixed restriction ρ_i such that, for a large enough $c > 0$, $L(F_i|_{\rho_i}) \leq c \cdot N^{1-\varepsilon} \cdot \mathsf{poly}(\log N)$ over $\log N$ variables. Thus, there exists a $\mathsf{GapAND}_{O(N)}\text{-}\mathcal{O}_{N^\beta}\text{-}\mathsf{Formula}[N^{1-\varepsilon} \cdot \mathsf{poly}(\log N)]$ formula over $\log N$ variables that computes f_y.

Now, observe that if f is computed by a $\mathsf{GapAND}_{O(N)}$-$\mathcal{O}_{N^\beta}$-formula of size $s(N)$, then there exists a (deterministic) $\mathcal{O}_{N^\beta}$-formula of size $s(N)$ which 0.9-approximates f. Using this fact, for every $y \in \{0,1\}^N$, there exists an $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}[N^{1-\varepsilon} \cdot \mathsf{poly}(\log N)]$ formula which 0.9-approximates f_y.

To finish the argument, observe that the number of $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}[N^{1-\varepsilon} \cdot \mathsf{poly}(\log N)]$ formulas over $\log N$ inputs is at most $2^{N^{1-\varepsilon+\beta} \cdot \mathsf{poly}(\log N)}$. Using an argument similar to that of Lemma 2.8, there exists a function over $\log N$ variables which cannot be 0.9-approximated by such formulas, which leads to a contradiction. □

We also localise the [CKLM19] technique for showing $\mathsf{MCSP}[2^n/n^4]$ lower bounds against cubic-sized formulas.

Proof of Item 3 Theorem 4.47. Firstly, we observe that an extension of the pseudorandom generator construction of [CKLM19] also works for formulas with an oracle top gate. We provide a brief sketch of this, as most of the proof remains the same.

Firstly, observe that pseudorandom shrinkage lemma from [IMZ19] (Lemma 4.8) extends directly to our case as well, which means under a suitable p-regular pseudorandom restriction samplable using $2^{O(\log^{2/3} s)}$ bits, any $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}$ of size s collapses to an $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}$ of size $2^{O(\log^{2/3} s)} \cdot p^2 \cdot s$, for any $s \geq N$ and $p \geq 1/\sqrt{s}$, with high probability over the restrictions. In particular, for $p = 1/s^{1/3}$, we get an $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}$ of size $s^{1/3} \cdot 2^{O(\log^{2/3} s)}$ with high probability.

Following the techniques of [CKLM19], we use this pseudorandom shrinkage lemma to construct a PRG of seed length $r = O(T + s^{1/3} \cdot 2^{O(\log^{2/3} s)})$ that 0.1-fools the class $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}[s]$, where T is the size of the encoding of an $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}$ which has shrunk under the application of the pseudorandom restriction. Moreover, their construction is such that the circuit complexity of the output of the generator (seen as a truth table) is just $O(r \cdot \mathsf{poly}(\log N))$.

Since any $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}[s]$ can be specified using $O(s \log s \cdot N^\beta)$ many bits, we have $T = O(s^{1/3} \cdot 2^{O(\log^{2/3} s)} \cdot N^\beta)$. Thus, for every $s \geq N$, there exists a PRG $G : \{0,1\}^r \to \{0,1\}^N$ that 0.1-fools $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}[s]$ where:

- The seed length $r = O(s^{1/3} \cdot 2^{O(\log^{2/3} s)} \cdot N^\beta)$.
- For every $z \in \{0,1\}^r$, there exists a circuit of size at most $O(s^{1/3} \cdot 2^{O(\log^{2/3} s)} \cdot N^\beta \cdot \mathsf{poly}(\log N))$ which takes input $j \in [\log N]$ and outputs the j^{th}-bit of $G(z)$.

In particular, when $s = N^{3-\varepsilon}$, the seed length is $r = N^{1-\Omega_{\beta,\varepsilon}(1)}$ and the circuit complexity of $G(z)$ is $O(N^{1-\Omega(1)})$.

Suppose that $\mathsf{MCSP}[2^n/n^4]$ can be computed by GapAND_M-$\mathcal{O}_{N^\beta}$-$\mathsf{Formula}[N^{3-\varepsilon}]$ formula C, where $M = O(N)$. Let $C_1, C_2, \ldots, C_M$ be the $\mathcal{O}_{N^\beta}$-$\mathsf{Formula}[N^{3-\varepsilon}]$ formulas which feed into the GapAND gate in C.

Firstly, observe that since C computes $\mathsf{MCSP}[s]$, we have that $\Pr_x\{C(x) = 1\} \leq o(1)$. By definition, we have that if $C(x) = 1$, then $C_i(x) = 1$ for all $i \in [M]$. On the other hand, if $C(x) = 0$, then $C_i(x) = 1$ with probability at most 0.1. Putting all this together, we have

$$\Pr_{i\in[M],x\in\{0,1\}^N}\{C_i(x) = 1\} \leq o(1) + 0.1 \leq 0.2$$

Furthermore, since $G(z)$ has circuit complexity $N^{1-\Omega(1)}$ for any $z \in \{0,1\}^r$, we have that $C(G(z)) = 1$ for every z. In particular, this means that $C_i(z) = 1$, for every $i \in [M], z \in \{0,1\}^r$.

$$\Pr_{i\in[M],z\in\{0,1\}^r}\{C_i(G(z)) = 1\} = 1$$

Using these equations together and averaging over $i \in [M]$, there exists an i such that

$$\left|\Pr_{x\in\{0,1\}^N}\{C_i(x) = 1\} - \Pr_{z\in\{0,1\}^r}\{C_i(G(z)) = 1\}\right| \geq 0.8$$

In other words C_i is distinguisher for G, which is a contradiction. □

Finally, we show that there is a language in E which cannot be computed by subcubic $\mathsf{GapAND}_{O(N)}$-formulas, but it *can* be computed in $O_{N^{o(1)}}$-$\mathsf{Formula}[N^2]$. Therefore, this lower bound does not localise in the sense of Theorem 4.47.

Theorem 4.48. *There is a language $L \in \mathsf{E}$, such that $L \notin \mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{3-\varepsilon}]$ for all constants $\varepsilon > 0$, but $L \in O_{N^{o(1)}}$-$\mathsf{Formula}[N^2]$.*

Proof. The function L is very similar to Andreev's function over $2N$ inputs (we assume N is a power of 2 for simplicity). For the second input to $\mathsf{Andreev}_{2N}$, we want to find a function $f_h : \{0,1\}^n \to \{0,1\}$ which cannot be 0.9-approximated by $N^{1-\varepsilon/2}$ formulas in $2^{O(N)}$ time (such a function exists by a simple counting argument as seen earlier). To find f_h, we simply enumerate all possible functions $f : \{0,1\}^n \to \{0,1\}$, and check whether it can be 0.9-approximated by an $N^{1-\varepsilon/2}$ size formula. There are $2^{2^n} = 2^N$ many functions on n bits and $(N^{1-\varepsilon/2})^{O(N^{1-\varepsilon/2})} = 2^{N^{1-\varepsilon/2}\cdot\mathsf{poly}(\log(N))}$ many formulas of size $N^{1-\varepsilon/2}$. Thus, a straightforward implementation of the algorithm runs in $2^{O(N)}$ time.

For the first input $x \in \{0,1\}^N$, L partitions x into $\log N$ blocks $x_1, x_2, \ldots, x_{\log N}$, of length $N/\log N$. L then computes $i \in \{0,1\}^{\log N}$ as $i = \mathsf{XOR}(x_1), \mathsf{XOR}(x_2), \ldots, \mathsf{XOR}(x_m)$. It then outputs $f_h(i)$. It is easy to verify that $L \in \mathsf{E}$ and $L \in O_{N^{o(1)}}$-$\mathsf{Formula}[N^2]$.

Now, suppose L can be computed by $\mathsf{GapAND}_{O(N)}$-formulas of size at most $N^{3-\varepsilon}$. If we apply a suitable random restriction keeping exactly one variable from each block

alive, we obtain a $\mathsf{GapAND}_{O(N)}$-$\mathsf{Formula}[N^{1-\varepsilon} \cdot \mathsf{poly}(\log(N))]$ formula for f_h (similar to Item 2 from Theorem 4.47). This implies that there is an $N^{1-\varepsilon} \cdot \mathsf{poly}(\log(N))$-size formula 0.9-approximating f_h, which is a contradiction. □

4.7 HM Frontier for $(n-k)$-Clique and AC^0

We first establish HM Frontier E in Section 4.7.1. Following this, we show that known lower bound techniques for AC^0 via random restrictions localise in Section 4.7.2.

4.7.1 Establishing HM Frontier E

Recall that we consider graphs on n vertices, described using an adjacency matrix of size $m = \Theta(n^2)$. An ℓ-clique is a subset of the vertices S such that each pair of vertices in S has an edge between them. The ℓ-clique problem is defined as: Given an input graph G over n vertices, does G have a clique of size at least ℓ?

We first prove the magnification result in HM Frontier E1.

Proposition 4.49. *Let $k(n) = (\log n)^C$, for some arbitrary $C \in \mathbb{N}$. If for every $d \geq 1$, there exists $\varepsilon > 0$ such that $(n-k)$-$\mathsf{Clique} \notin \mathsf{AC}^0_d[m^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.*

Proof. The proof follows by a reduction to a similar magnification theorem by [OS18a] (Theorem 7): If for every $d \geq 1$, there exists $\varepsilon > 0$ such that k-$\mathsf{Vertex\text{-}Cover} \notin \mathsf{AC}^0_d[m^{1+\varepsilon}]$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

Any graph G on n vertices has a clique of size $\geq n-k$ iff its complement graph $\overline{G}$ has an independent set of size $\geq n-k$. In turn, the latter statement holds iff $\overline{G}$ has a vertex cover of size $\leq k$. Thus, the reduction takes the negation of its input literals and checks if it has a vertex cover of size $\leq k$. In other words, the AC^0-complexities of $(n-k)$-Clique and k-$\mathsf{Vertex\text{-}Cover}$ are the same. Thus, the lower bound for $(n-k)$-Clique implies the magnification theorem for k-$\mathsf{Vertex\text{-}Cover}$ by [OS18a]. □

Under the Exponential Time Hypothesis (ETH), it is known that k-$\mathsf{Vertex\text{-}Cover}$ cannot be computed in time $2^{o(k)}\mathsf{poly}(m)$ (see [IPZ01], Theorem 29.5.9 in [DF13]). Thus, if $k = \omega(\log n)$, it is plausible that k-vertex cover problem is not in P. The proof of Proposition 4.49 implies that the complexities of $(n-k)$-Clique and k-$\mathsf{Vertex\text{-}Cover}$ for non-monotone computations are equivalent. As a consequence, we observe Item E2 of the frontier using a similar hypothesis for non-uniform computations.

Proposition 4.50. *Suppose that ETH for non-uniform circuits holds. Then $(n-k)$-$\mathsf{Clique} \notin \mathsf{P/poly}$, for $\omega(\log n) \leq k \leq n - \omega(\log n)$.*

Further discussion on the conditional hardness of k-Vertex-Cover that also applies to $(n-k)$-Clique can be seen in [OS18a].

Item E3 is well-known [Ajt83, FSS84, Yao85, Hås86]. The next proposition implies unconditional monotone circuit lower bounds for detecting large cliques in E4.

Proposition 4.51 ([AJ08], Section 9.2 in [Juk12]). *For any $k \leq n/2$, every monotone circuit computing $(n-k)$-Clique requires $2^{\Omega(k^{1/3})}$ gates.*

Interestingly, the problem can be solved by (bounded depth) polynomial size monotone circuits if $k \leq \sqrt{\log n}$ [AJ08].

4.7.2 Locality Barrier for AC^0 Lower Bounds Above the Threshold

This section captures an instantiation of the locality barrier for HM Frontier E.

Theorem 4.52 (Locality Barrier for HM Frontier E). *The following results hold.*

(E1$^{\mathcal{O}}$) (Oracle Circuits from Magnification) *There exists an oracle $\mathcal{O}$ such that, for each $k = (\log n)^C$ and every large enough depth d, $(n-k)$-Clique $\in (\mathsf{AC}^0_d)^{\mathcal{O}}[m^{1+\varepsilon_d}]$, where $\varepsilon_d \to 0$ as $d \to \infty$, and the corresponding circuit employs a single oracle gate $\mathcal{O}$ of fan-in at most $O((\log n)^{4C})$.*

(E3$^{\mathcal{O}}$) (Extension of Lower Bound Techniques) $\mathsf{Parity} \notin (\mathsf{AC}^0)^{\mathcal{O}}[\mathsf{poly}(n)]$ *if the total number of input wires in the circuit feeding the $\mathcal{O}$-gates is $n/(\log n)^{\omega(1)}$.*

Proof. The first item follows by inspection of the proof of Proposition 4.49. This in turn uses the oracle circuit construction from [OS18a]. Recall that the circuit in [OS18a] simulates the well-known kernelisation algorithm for vertex cover highly efficiently. This algorithm takes a graph G over n vertices and produces a smaller instance H over $O(k^2)$ vertices, with a parameter $k' \leq k$ such that H has a k'-sized vertex cover iff G has k-sized vertex cover. Thus, the construction passes an adjacency matrix description of H of size $O(k^4)$ and k' as inputs to an oracle $\mathcal{O}$ which computes k'-vertex cover, and outputs $\mathcal{O}$'s answer on this. The correctness of the construction follows from the transformation of an $(n-k)$-clique instance to that of k-vertex cover in Proposition 4.49, and consequently the equivalences which imply that H has a k'-sized vertex cover iff G has an $(n-k)$-clique.

The second item follows from the simulation of oracle circuits via interactive compression games (see Proposition 5.1 in [OS15]) and lower bounds on the communication cost for the latter. They show that one can view an oracle circuit as an interactive protocol between 2 parties, where one party is restricted to computation within a fixed circuit class and the other has unbounded computational power. The total fan-in of all the oracle gates

in the circuit is the number of bits communicated to the party with unbounded power. Using this connection with the main result from [CS12], which uses random restrictions to show that any AC^0-compression game (restricted party can only make computations in AC^0) for Parity requires at least $n/(\log n)^{\omega(1)}$ bits to be communicated, we observe the desired lower bound. □

Informally, the main difficulty with using random restrictions in HM Frontier E, is that as soon as the Boolean circuit gets simplified so that the oracle gate $\mathcal{O}$ is only fed by input literals, just setting the $O(\log n)^{4C}$ literals eliminates the oracle gate. Sacrificing such a small number of variables wont affect a typical worst-case lower bound based on the random restriction method.

4.8 Locality Barrier for Lower Bounds Below the Threshold

The localisations presented in this section show that one cannot obtain strong circuit lower bounds by "lowering the threshold" in certain hardness magnification proofs. In other words, the localisations of the lower bound techniques in this section rule out a family of potential approaches for establishing strong lower bounds by magnifying an *already known* (weak) lower bound for a variant of MCSP or MKtP. As a consequence of one of our results (Theorem 4.59 in Section 4.8.2), we also refute the Anti-Checker Hypothesis from [OPS19].

4.8.1 Locality barrier for AC^0 lower bounds via pseudorandom restrictions

Consider the following hypothesised approach to show that $\mathsf{NP} \not\subseteq \mathsf{NC}^1$: Construct a sequence of oracle AC^0-circuits of linear size which compute $\mathsf{MKtP}[\log^4 N, 2\log^4 N]$, where the oracle gates access some $\mathcal{O} \in \mathsf{NP}$ and have fan-in $\mathsf{poly}(\log N)$. Existence of such circuits implies that $\mathcal{O}$ cannot be computed by formulas of size n^k, for any $k > 0$. Indeed, if this was not true, then we could first use Lemma 4.14 to get an AC^0_d simulation for the NC^1 circuit computing $\mathcal{O}$ for a large enough d and then, replace each oracle in C with this circuit, to get a constant depth circuit of size $N^{1+o(1)}$ which computes $\mathsf{MKtP}[\log^4 N, 2\log^4 N]$. This would contradict the MKtP lower bound against AC^0 in Theorem 4.15.

We show that the pseudorandom restriction method used to show lower bounds for MCSP or MKtP against AC^0 (in [CKLM19] and Theorem 4.15) can be localised. This refutes the existence of such AC^0 circuit constructions using local oracles for MKtP and

MCSP, even when the oracles are arbitrary. Consequently, this excludes magnification theorems which can be proved by approaches that unconditionally give such oracle circuit constructions. The main result of the section is the localisation of Theorem 4.15.

Theorem 4.53. *For every constant $d > 1$, and $O_1, \ldots, O_d \in \mathbb{N}$ such that $\prod_{i=1}^{d} O_i \leq N/\log^{6d} N$, $\mathsf{MKtP}[\log^6 N, 2\log^6 N]$ cannot be computed by oracle AC^0-circuits of polynomial size and depth d, where the oracle fan-in in the i^{th} layer is O_i.*

A similar result extends for the MCSP lower bound in [CKLM19] (Theorem 3) as well.

Remark 4.54. *We remark that the constraint on oracles in the above theorem is incomparable to the second item of 4.52. Here we focus on the maximum oracle fan-in at each level, while the focus there is on the total fan-in of all oracles. A lower bound result for an explicit problem with parameters similar to Theorem 4.53 is not known for $\mathsf{AC}^0[2]$ oracle circuits (see Section 5 in [OS15] for results in this direction).*

For the localisation of the lower bound, we use a k-wise independence generator to sample pseudorandom restrictions in the derandomised switching lemma.

We first prove a version of derandomised switching lemma for functions which are computed by feeding the outputs of m shallow decision trees into an arbitrary oracle.

Lemma 4.55. *Let $\mathcal{O} : \{0,1\}^m \to \{0,1\}$ be an arbitrary function and $D_1, \ldots, D_m$ be decision trees of depth r over variables $x_1, \ldots, x_N$. Define $F(x) = \mathcal{O}(D_1(x), \ldots, D_m(x))$. Then, for every $t \in \mathbb{N}$ and for every $r(t+1)$-wise independent p-regular pseudorandom restriction $\mathcal{D}$,*

$$\Pr_{\rho \sim \mathcal{D}}\{\mathsf{DT}_{depth}(F|_\rho) > t\} \leq \left(\frac{2mpe^2 r}{t+1}\right)^{t+1}$$

Proof. Take any fixed restriction $\rho : [N] \to \{0,1,*\}$. To analyse $\mathsf{DT}_{\text{depth}}(F|_\rho)$, we direct our focus to the decision tree $\widetilde{T}|_\rho$ which computes $F|_\rho$ as defined below.

- $\widetilde{T}|_\rho$ first runs $D_1|_\rho$. In other words, for any variable x_j queried by D_1, if $\rho_j \in \{0,1\}$, then $D_1|_\rho$ proceeds by setting $x_j = \rho_j$ and otherwise, it queries x_j on both values. At the end of each decision tree path τ_1, $\widetilde{T}|_\rho$ starts the simulation of $D_2|_{\rho\tau_1}$.

- Continuing in this fashion, for every $2 \leq i \leq m$, at the end of each decision tree path τ_i in the simulation of $D_i|_{\rho,\tau_1,\ldots,\tau_{i-1}}$, $\widetilde{T}|_\rho$ proceeds by simulating D_{i+1} under the restriction $\rho, \tau_1, \ldots, \tau_i$.

- At the end of the simulation, for any path $\mathcal{P}$ in the tree $\widetilde{T}|_\rho$, let $a_\mathcal{P} = (a_1, \ldots, a_m)$ be the outputs of $D_1|_\rho, \ldots, D_m|_\rho$ obtained by setting variables according to the path $\mathcal{P}$. At the end of $\mathcal{P}$, the leaf is set to the value $\mathcal{O}(a_\mathcal{P})$.

Let $\mathsf{depth}(\widetilde{T}|_\rho)$ be the decision tree depth of $\widetilde{T}|_\rho$. Clearly, $\mathsf{DT}_{\mathrm{depth}}(F|_\rho)$ is at most $\mathsf{depth}(\widetilde{T}|_\rho)$ and it is enough for us to upper bound $\Pr\{\mathsf{depth}(\widetilde{T}|_\rho) > t\}$.

Observe that $\mathsf{depth}(\widetilde{T}|_\rho) > t$ iff there exists a path z of length $t+1$ which occurs as the prefix of some computational path in $\widetilde{T}|_\rho$. For any $z \in \{0,1\}^{t+1}$, define A_z as this event.

$$\begin{aligned}\Pr_{\rho\sim\mathcal{D}}\{\mathsf{depth}(\widetilde{T}|_\rho) > t\} &= \Pr_{\rho\sim\mathcal{D}}\{\exists z \in \{0,1\}^{t+1} \text{ such that } A_z\} \\ &\leq \sum_{z\in\{0,1\}^{t+1}} \Pr_{\rho\sim\mathcal{D}}\{A_z\}\end{aligned}$$

where the second inequality holds by the union bound.

Fix any $z \in \{0,1\}^{t+1}$. For $\rho \sim \mathcal{D}$, consider the computation path traversed by $\widetilde{T}|_\rho$ when its first $(t+1)$ queries are answered according to z. This implies that there exists $w \leq t+1$ and decision trees $D'_1, \ldots, D'_w$, such that $\widetilde{T}|_\rho$ makes its first $(t+1)$ queries in these trees and answers according to z.

Now, for any such $w \leq t+1$ and $D'_1, \ldots, D'_w$, if $\widetilde{T}|_\rho$ were to be simulated only on $D'_1, \ldots, D'_w$ by answering according z, its depth would still be at least $t+1$. Since $\mathcal{D}$ is $r(t+1)$-wise independent, the probability that this happens is at most

$$p^{t+1}\binom{\dot{wr}}{t+1} \leq \left(\frac{pewr}{t+1}\right)^{t+1} \leq (per)^{t+1}$$

Using a union bound over all possibilities, we have

$$\begin{aligned}\Pr_{\rho\sim\mathcal{D}}\{\mathsf{depth}(\widetilde{T}|_\rho) > t\} &= \sum_{z\in\{0,1\}^{t+1}} \Pr_{\rho\in\mathcal{D}}\{A_z\} \\ &\leq \sum_{z\in\{0,1\}^{t+1}} \sum_{w=1}^{t+1} \binom{m}{w} \cdot (per)^{t+1} \\ &\leq 2^{t+1} \cdot \left(\frac{me}{t+1}\right)^{t+1} \cdot (per)^{t+1} = \left(\frac{2mpe^2r}{t+1}\right)^{t+1}\end{aligned}$$

□

We next need the following results on k-wise independent distributions fooling CNFs.

Lemma 4.56 ([Baz09, Tal17b])**.** *Any $O\left(\log\left(\frac{S}{\varepsilon}\right) \cdot \log S\right)$-wise independent distribution ε-fools CNFs (or DNFs) of size S.*

We now prove the main result of the section.

Proof of Theorem 4.53. Towards a contradiction, let C be an oracle AC^0-circuit on N inputs of size S and depth d computing $\mathsf{MKtP}[\log^6 N, 2\log^6 N]$, where the oracle fan-in

in the i^{th} layer is O_i and $S = N^c$ for any constant $c > 0$. For each $i \in [d]$, let S_i be the number of gates in layer i in C.

Define $t = 3c \log N, \varepsilon_0 = 2^{-32c^2 \log^2 N}$ and $p = 2^{-q} = 1/120c \log N = 1/40t$. Let $\mathcal{D}$ be the k-wise independent p-regular pseudorandom restriction which is obtained from Lemma 4.8, where $k = O\left(\log\left(\frac{S \cdot 2^t \cdot 2^{t(q+1)}}{\varepsilon_0}\right) \cdot \log\left(S \cdot 2^t \cdot 2^{t(q+1)}\right) \cdot q\right) = \widetilde{O}(\log^3 N)$. The seed length for sampling $\mathcal{D}$ is $r = O(k \cdot q \cdot \log N) = \widetilde{O}(\log^4 N)$.

The proof is very similar to that of Lemma 4.19 and proceeds in d iterations. In the i^{th} iteration, we apply a composition of $m_i = \log_{1/p} O_i + 1$ independently sampled pseudorandom restrictions drawn from $\mathcal{D}$, to the circuit C. This composition denoted by ρ_i, is a k-wise independent p_i-regular pseudorandom restriction, where $p_i = p^{m_i} = p/O_i$. The proof proceeds by maintaining the invariant that with high probability, at the end of the i^{th} iteration the functions computed by the gates in level i can be computed by decision trees of depth t (at the beginning, we treat the input variables as decision trees of depth 1).

In iteration i, assume that after the previous iterations of restrictions, each input to the gates at level i can be computed by a decision tree of depth t. We have the following cases:

- Since $k > t(t+1)$, we apply Lemma 4.55 to see that under the application of ρ_i, every oracle gate in level i whose inputs are depth t decision trees can be computed by a decision tree of depth larger than t, with probability at most

$$S_i \left(\frac{2O_i p_i e^2 t}{t+1}\right)^{t+1} \leq S_i \left(2pe^2\right)^{t+1} \leq S_i \left(\frac{1}{N^{2c}}\right)$$

- The function computed by any AND/OR gate ϕ in level i whose inputs are depth t decision trees, is equivalent to a CNF/DNF of width t and size at most $S \cdot 2^t$. From Lemma 4.56, observe that $\mathcal{D}$ ε_0-fools CNFs/DNFs of size $S \cdot 2^t \cdot 2^{t(q+1)}$. Using this within Lemma 4.17 (derandomised switching lemma for CNFs/DNFs), the probability that $\phi|_{\rho_i}$ can be computed by a decision tree of depth greater than t is

$$\begin{aligned}
&\leq S_i \cdot \left(2^{2t+1}(5pt)^t + \varepsilon_0 \cdot 2^{(t+1)(3t+\log S)}\right) \\
&\leq S_i \cdot \left(2^{2t+1}(5t/40t)^t + \varepsilon_0 2^{(3c\log N+1)(10c\log N)}\right) \\
&\leq S_i \cdot \left(2^{1-t} + \varepsilon_0 2^{31c^2 \log^2 N}\right) \leq S_i \cdot \left(\frac{1}{N^{2c}}\right)
\end{aligned}$$

  Note that, although we refer to ϕ being restricted by ρ_i, we only use one of the m_i pseudorandom restrictions.

Define the restriction ρ as the composition of $\rho_1, \ldots, \rho_d$. Using a union bound, the probability that $C|_\rho$ shrinks to a depth t decision tree, is at least $1 - 2S \cdot N^{-2c}$. When $S = N^c$, with probability at least $3/4$ over ρ, C_ρ can be computed by a depth t decision tree.

Let m be the total number of pseudorandom restrictions applied to C. Now, $m = \sum_i^d m_i = d + \log_{1/p}\left(\prod_{i=1}^d O_i\right) \leq \log_{1/p} N - 3d$. Thus, the probability p^* that any variable is not restricted is $p^* = p^m \geq p^{-3d}/N$. Since $k \geq 2$, ρ is also a p^*-regular pairwise independent distribution. Thus, using Chebyshev's inequality (Lemma 4.6), the probability that at least $p^*N/2$ variables are left unrestricted after the application of all the restrictions is at least $3/4$. Put together, this implies the existence of ρ such that $C|_\rho$ can be computed by a decision tree of depth t and at least $p^*N/2$ variables are left unrestricted.

We now construct a YES instance w^Y and a NO instance w^N such that $C(w^Y) = C(w^N)$ and this contradicts the fact that C computes $\mathsf{MKtP}[\log^6 N, 2\log^6 N]$. Firstly, define $w^Y \in \{0,1\}^N$ as $\rho \circ 0^N$. Since the seed length is $r \cdot m = \widetilde{O}(\log^5 N)$ for N large enough, and there exists an algorithm that computes each bit of ρ in time $\mathsf{poly}(r) \cdot m$ (cf. Lemma 4.8), we see that w^Y is a YES instance, i.e. $\mathsf{Kt}(w^Y) = \widetilde{O}(\log^5 N) \leq \log^6 N$. On the other hand, let W be the set of variables queried by the decision tree of depth t computing $C|_\rho$. The number of assignments over $\rho^{-1}(*) \setminus W$ is at least $2^{O(\log^{3d} N)/2 - O(\log N)} \geq 2^{2\log^6 N + 1}$. This implies the existence of an assignment $h \in \{0,1\}^{\rho^{-1}(*)\setminus W}$, such that any string in $\{0,1\}^N$ which agrees with h over $\rho^{-1}(*) \setminus W$ has Kt complexity larger than $2\log^6 N$. w^N is extended from ρ by fixing every variable in W to be 0 and the remaining variables to be h, to get a NO instance. Now, since w^Y and w^N agree on $\rho^{-1}(\{0,1\}) \cup W$ and C becomes a constant when set according to this assignment, $C(w^Y) = C(w^N)$. □

4.8.2 The Sub-Quadratic Formula Lower Bound of [HS17]

In this section, we prove that the nearly quadratic formula size lower bound of [HS17] localises, and thereby proving the third item of Theorem 4.1. By localising [HS17], we exclude a family of magnification theorems obtained by approaches that unconditionally produce formulas with oracles, and essentially address a question from [OPS19]. It also suggests that the consideration of almost-formulas in HM Frontier C (Section 4.5) is unavoidable.

Before stating the main theorem of this section, we need to establish the following complexity measure on oracle formulas.

Definition 4.57 (Size Measure for Oracle Formulas). *For any integer $t > 1$ and any oracle formula F with oracle gates to $\mathcal{O}$ having fan-in B, define* $\mathsf{Formula}_t(F)$ *as follows:*

$$\mathsf{Formula}_t(F) = \begin{cases} \mathsf{Formula}_t(F_1) + \mathsf{Formula}_t(F_2), & \text{if } F = F_1 \wedge F_2 \text{ or } F_1 \vee F_2 \\ B^{t-1} \cdot \left(\sum_{i=1}^{B} \mathsf{Formula}_t(F_i) \right), & \text{if } F = \mathcal{O}(F_1, \ldots, F_B) \end{cases}$$

We say that a function $f \in \mathsf{Formula}_t^{\mathcal{O}}[s]$, if there exists an integer $B = B(N)$ and a sequence of oracle formulas $\{F_N\}$ with oracle gates to $\mathcal{O}$ having fan-in B, such that F_N computes f_N for every N large enough, and $\mathsf{Formula}_t(F_N)$ *is at most $s(N)$.*

Informally, for any oracle formula F having oracle gates to $\mathcal{O}$ with fan-in B, $\mathsf{Formula}_t(F)$ is the size of the De Morgan formula obtained by replacing every oracle gate in F with an equivalent formula of size B^t, which reads each of its B inputs exactly B^{t-1} times.

One plausible approach for proving strong circuit lower bounds is to show a highly-efficient small fan-in oracle formula construction for $\mathsf{MCSP}[2^{\sqrt{n}}]$ and lower the magnification threshold to known sub-quadratic formula lower bounds for $\mathsf{MCSP}[2^{\sqrt{n}}]$ by [HS17]. This approach can be formalised as:

Proposition 4.58. *For any $t > 0$, if $\mathsf{MCSP}[2^{\sqrt{n}}] \in \mathsf{Formula}_{t+1}^{\mathsf{NP}}[N^{2-\varepsilon}]$ for some $\varepsilon > 0$ (formulas which have oracle access to some language in* NP*), then $\mathsf{NP} \not\subseteq \mathsf{Formula}[n^t]$.*

Proof. Suppose that $\mathsf{NP} \subseteq \mathsf{Formula}[n^t]$. Let $\{F_N\}$ be the sequence of NP-oracle formulas computing $\mathsf{MCSP}[2^{\sqrt{n}}]$. By adding enough dummy leaves, any language in NP can be computed by formulas of size n^{t+1} which reads each input exactly n^t times. Replacing every NP-oracle gate in F with these formulas, we get a formula of size $N^{2-\varepsilon}$ that computes $\mathsf{MCSP}[2^{\sqrt{n}}]$, which contradicts the [HS17] formula size lower bound. □

In particular, Proposition 4.58 implies that if $\mathsf{MCSP}[2^{\sqrt{n}}] \in \mathsf{Formula}_{t+1}^{\mathsf{NP}}[N^{2-\varepsilon}]$ for every $t > 0$, then $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

Our main result in this section rules out the existence of such magnification approaches even when $t = 3$, by localising the MCSP lower bound by [HS17], with a mild constraint on the adaptivity of the oracle circuits (maximum number of oracles along any path in the formula). Of course, this refutes the existence of such oracle formula constructions for $\mathsf{MCSP}[2^{\sqrt{n}}]$ for any $t \geq 4$.

Theorem 4.59 (Localising the [HS17] lower bound). *There exists a universal constant c, such that for all constants $\varepsilon > 0$ and every oracle $\mathcal{O}$, $\mathsf{MCSP}[n^c, 2^{\varepsilon n/3}] \notin \mathsf{Formula}_3^{\mathcal{O}}[N^{2-\varepsilon}]$, where the adaptivity of the oracle formulas is at most $o(\log N / \log\log N)$.*

In other words, $\mathsf{MCSP}[2^{\sqrt{n}}]$ cannot be computed by oracle formulas F, such that $\mathsf{Formula}_3(F) \leq N^{2-\varepsilon}$ and adaptivity of F is at most $o(\log N / \log\log N)$.

Remark 4.60. *Note that for any F, under the assumption that $\mathsf{Formula}_3(F) \leq N^{2-\varepsilon}$, the adaptivity of F can be at most $O(\log N)$.*

One important consequence of Theorem 4.59 is that it refutes the Anti-Checker Hypothesis from [OPS19].

The Anti-Checker Hypothesis [OPS19] *For every $\lambda \in (0,1)$, there exists $\varepsilon > 0$ and a collection $\mathcal{Y} = \{Y_1, \ldots, Y_\ell\}$ of sets $Y_i \subseteq \{0,1\}^n$, where $\ell = 2^{(2-\varepsilon)n}$ and each $|Y_i| = 2^{n^{1-\varepsilon}}$, for which the following holds.*

If $f : \{0,1\}^n \to \{0,1\}$ and $f \notin \mathsf{SIZE}[2^{n^\lambda}]$, then some set $Y \in \mathcal{Y}$ forms an anti-checker for f, i.e. for each circuit C of size $2^{n^\lambda}/2n$, there is an input $y \in Y$ such that $C(y) \neq f(y)$.

Corollary 4.61. *The Anti-Checker Hypothesis is false.*

Proof. [OPS19] show that the Anti-Checker Hypothesis implies that for some $\varepsilon > 0$, $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}]$ can be solved by an oracle formula F of size $N^{2-\varepsilon}$, with $N^{2-\varepsilon}$ many NP-oracle gates of fan-in $\mathsf{poly}(n) \cdot 2^{n^{1-\varepsilon}} = N^{o(1)}$, which are present only in the layer above the leaves. In particular, this implies that $\mathsf{Formula}_3(F_N) \leq N^{2-\varepsilon+o(1)}$ and $\mathsf{MCSP}[2^{n^{1/3}}, 2^{n^{2/3}}] \in \mathsf{Formula}_3^{\mathsf{NP}}[N^{2-\varepsilon+o(1)}]$. This leads to a contradiction, as such oracle formulas are ruled out by Theorem 4.59. □

We are now ready to prove Theorem 4.59. We first give a high level overview of the proof. The argument is similar to [HS17, OPS19], who show sub-quadratic formula lower bounds for MCSP (and MKtP) using an iterative restriction process. Suppose there exists an oracle formula F over N inputs, such that $\mathsf{Formula}_3(F) \leq N^{2-\varepsilon}$ and adaptivity $o(\log N / \log\log N)$ that computes $\mathsf{MCSP}[n^c, 2^{\varepsilon n/3}]$.

For k small enough ($k = \mathsf{poly}(\log N)$), we apply $r \approx \log_k(N^{2-\varepsilon})$ many pseudorandom restrictions $\rho_1, \ldots, \rho_r$ on F, which are independently sampled from a $\frac{1}{\sqrt{k}}$-regular k-wise independent distribution $\mathcal{D}$, given by Lemma 4.8.

To analyse the size of F under the influence of $\rho_1, \ldots, \rho_r$, we define a potential function Φ. For any oracle formula, Φ is defined in a bottom-up manner, building on inductive blocks which we call as *maximal sub-formulas*. The potential Φ of an oracle formula is a random variable which will be defined such that it is not just a function of the structure of the formula, but also is dependent on the pseudorandom restrictions which have been applied so far.

Crucially, for any oracle formula F, we require Φ to satisfy the following properties:

1. When F is hit by a p-regular pseudorandom restriction from $\mathcal{D}$, Φ shrinks by a factor close to p^2 in expectation (note that $p = 1/\sqrt{k}$).

2. With high probability, for every iteration of the restriction, and every maximal sub-formula G of F, if $\Phi = 0$, then G has shrunk "significantly", i.e. depends on "very few" variables. This notion of high shrinkage depends on the structure of G and will be defined more precisely soon.

3. Either $\Phi = 0$ or $\Phi \geq 1$. From the second item, this implies that if F has not shrunk significantly, then $\Phi(F) \geq 1$.

We use a case-based analysis based on the structure of the oracle formula to inductively define Φ which satisfies these properties. In particular, $\Phi(F)$ is defined in such a way that it is closely related to $\mathsf{Formula}_3(F)$ and thus, is at most $N^{2-\varepsilon+o(1)}$ at the beginning (in fact, as we shall soon see, the motivation for defining the measure $\mathsf{Formula}_3(F)$ is based on our inductive definition of $\Phi(F)$).

Using the properties of Φ, we show that after roughly r rounds of restrictions from $\mathcal{D}$, $\Phi(F)$ becomes 0 with high probability because of its shrinkage under pseudorandom restrictions. This means that F depends on very few variables (say $\mathsf{poly}(\log N)$), with high probability. Further, we get a final composed restriction which has a low circuit complexity. On the other hand, we show that the composition of the r restrictions leaves a $1/N^{\Omega(1)}$-fraction of the variables unrestricted with high probability. As seen earlier, we can easily extend the composed restriction to a YES instance w^Y and a NO instance w^N, such that F accepts them both. This contradicts the fact that F computes $\mathsf{MCSP}[n^c, 2^{\varepsilon n/3}]$.

Before diving into the proof, we need the following lemmas from [IMZ19].

Lemma 4.62 (Lemma 4.3 in [IMZ19], [Tal14])**.** *For any positive k and any formula F on a set of variables $x_1, \ldots, x_N$ with $L(F) \geq k$, there exists at most $6L(F)/k$ formulas G_i with $L(G_i) \leq k$, where the G_i may consist of some variables in addition to the original input variables of F (called special variables), such that the following holds: For any restriction ρ, $L(F|_\rho) \leq \sum_i L(G_i|_{\rho'})$, where $\rho'(x_j) = \rho(x_j)$ for $x_j \in \{x_1, \ldots, x_N\}$, and $\rho'(j) = *$ otherwise. Moreover, each G_i depends on at most 2 special variables.*

The special variables here refer to the roots of other sub-formulas in the collection which were at the bottom of G_i in the original formula F.

Since the number of special variables S is just 2, we can use a constant-sized formula for the addressing function over S, to construct a formula computing $G_i|_\rho$ over all possible assignments to S. This leads us to the following shrinkage under expectation result for k-wise independent pseudorandom restrictions.

Lemma 4.63 (Lemma 27 in [HS17] (based on [IMZ19, KRT17]))**.** *Let F be any formula over N inputs and $k \in \mathbb{N}$ such that $L(F) \geq k$. If $\mathcal{D}$ is a $(1/\sqrt{k})$-regular k-wise independent distribution, then $\mathbf{E}_{\rho \in \mathcal{D}}[L(F|_\rho)] \leq 72 \cdot L(F)/k$.*

Proof of Theorem 4.59. Let F_N be an oracle formula, with oracle access to some function $\mathcal{O}$. Let $k = \log^3 N$ and let $\mathcal{D}$ be a $\left(\frac{1}{\sqrt{k}}\right)$-regular k-wise independent distribution over restrictions on N variables, from Lemma 4.8. For r will be set later, we apply r independent pseudorandom restrictions $\rho_1, \ldots, \rho_r$, each identically distributed according to $\mathcal{D}$, to the oracle formula F.

Define a *maximal sub-formula* of F as any oracle sub-formula of F which is either (a) rooted at an oracle gate (b) rooted at a gate whose parent is an oracle gate, or (c) the formula F itself. For any maximal sub-formula G of F and for any integer $0 \leq i \leq r$, define $\Phi(G, \rho_1, \ldots, \rho_i)$ as a real-valued random variable, which defines the *potential function* of G after the first i pseudorandom restrictions $\rho_1, \ldots, \rho_i$. Maximal sub-formulas form the inductive blocks used in the construction of $\Phi(F)$. Note that, $\Phi(F)$ and $\Phi(F, \rho_1, \ldots, \rho_r)$ are the potentials of F at the beginning and the end of the restriction process.

Next, we say that a sub-formula G is "*tiny*" if (a) the top gate is an oracle and G depends on at most $\log N$ variables, or else (b) it depends on at most $c_{\text{tiny}} \cdot k$ number of variables, for some constant c_{tiny} we define later. These parameters ensure that in the case analysis for the inductive construction of $\Phi(F)$, any maximal sub-formula becomes tiny with high probability, under the pseudorandom restriction.

Now, recall the *properties required* from Φ. For any oracle formula F,

1. $\mathbf{E}_{\rho \sim \mathcal{D}}[\Phi(F|_\rho)] \leq \frac{c_F}{k} \cdot \Phi(F)$, where c_F is a constant which depends on F, but is upper bounded universally by a constant.

2. With probability at least $1 - p_F$, for every iteration $1 \leq i \leq r$ and every maximal sub-formula G of F, if $\Phi(G, \rho_1, \ldots, \rho_i) = 0$, then G is tiny, where $p_F \in [0, 1]$ depends on F and is upper bounded by N^{-2}.

   Furthermore, $\Phi(G, \rho_1, \ldots, \rho_i)$ is either 0 or at least 1.

We next construct the *potential function* Φ. For any oracle formula F, let G be any maximal sub-formula of F. Consider the following cases for G.

Case I: G is a Pure Formula

This forms the simplest case and we can directly use known techniques here. Suppose G is a pure formula (formulas with no oracles) of size S. Then,

$$\Phi(G) = \begin{cases} S & \text{if } S > 80k, \\ 0 & \text{otherwise.} \end{cases}$$

We now see that $\Phi(G)$ easily satisfies the properties. Indeed, the first property is satisfied because of Lemma 4.63. Thus, when $S \geq 80k$, the expected size of G under $\rho \sim \mathcal{D}$ drops by a factor k/c_G ($c_G = 72$). The second and third properties are easily satisfied by the definition, by setting $p_G = 0$ and c_{tiny} to be larger than 80.

Case II: G is an Oracle Gate

Suppose G is a pure oracle gate $\mathcal{O}$ with fan-in T, i.e. each input to $\mathcal{O}$ is just an input variable. We set the potential as:

$$\Phi(G) = \begin{cases} T^2k^3 \text{ if } T > \log N, \\ 0 \qquad \text{otherwise.} \end{cases}$$

After each restriction ρ, we set $\Phi(G|_\rho)$ to $\Phi(G)/k$, with $c_G = 1$. In other words, $\Phi(G, \rho_1, \ldots, \rho_i)$ is set to $\Phi(G)/k^i$. Furthermore, if for some $1 \leq i \leq r$, $\Phi(G, \rho_1, \ldots, \rho_i)$ is less than 1, then $\phi(G, \rho_1, \ldots, \rho_i)$ is set to 0 henceforth. The definition of Φ ensures the first property.

To show the second item, observe that $\Phi(G)$ becomes 0 after at least $R = 2\log_k T$ many restrictions.[19] The composition of these restrictions is a k-wise independent restriction which keeps a variable unrestricted with probability at most $1/T$. Thus, when $\Phi(G, \rho_1, \ldots, \rho_R)$ becomes 0, the probability that the restricted formula depends on more than $\log N$ variables is at most :

$$\binom{T}{\log N} \cdot \left(\frac{1}{T}\right)^{\log N} \leq \left(\frac{e}{\log N}\right)^{\log N} \leq \frac{1}{N^4}$$

Setting $p_G = 1/N^4$, we see that with probability at least $1 - p_G$, $\Phi(G, \rho_1, \ldots, \rho_R) = 0$ implies that $G|_{\rho_1,\ldots,\rho_R}$ is tiny.

Case III: G is a Formula with a Top Oracle Gate

Consider the case where G is an oracle formula with a top oracle gate of fan-in T. Let $G_1, \ldots, G_T$ be the maximal sub-formulas feeding into the oracle gate and let $\Phi(G_j)$ be their respective potentials. Note that Case II is a special case.

We proceed with the process of applying restrictions iteratively. At the beginning, the potential of G is set as:

$$\Phi(G) = \max\left(\sum_{j=1}^{T} \Phi(G_j), 1/k\right) \cdot T^2 \cdot k^4$$

Whenever $\sum_{j=1}^{T} \Phi(G_j)$ first becomes 0, say after the i^{th} restriction, the potential $\Phi(G, \rho_1, \ldots, \rho_i)$ is set to T^2k^3. Note that, this holds even when $i = 0$, i.e. $\Phi(G_j) = 0$ for

[19] For our purposes, it is enough if $\Phi(G) = T^2$, if $T > \log N$. The potential used above is to maintain consistency with Case III.

all $1 \leq j \leq T$ to begin with. Going further, we reduce the potential by a factor of k after each restriction until it becomes less than 1, beyond which it is set to 0 every time. Note that c_G is set to $\max_{1 \leq j \leq T} c_{G_j}$.

The first property holds for Φ by the inductive nature of its construction. Indeed, suppose that each $\Phi(G_j)$ reduces by a factor of k/c_{G_j} in expectation, after each restriction from $\mathcal{D}$. When $\sum_{j=1}^{T} \Phi(G_j) > 0$, we see that after each restriction, $\Phi(G)$ drops by a factor at most $k/\max_j c_{G_j} = k/c_G$ in expectation. When $\sum_{j=1}^{T} \Phi(G_j) = 0$, our definition of $\Phi(G)$ ensures this property holds. Finally, for the case where $\sum_{j=1}^{T} \Phi(G_j) > 0$ before the restriction and 0 after, one can see that Φ drops from at least T^2k^4 to at most T^2k^3.

Next, suppose that $\Phi(G)$ becomes 0 after R restrictions from $\mathcal{D}$, for some $R \in \mathbb{N}$. Observe that this happens only when $\sum_{j=1}^{T} \Phi(G_j)$ becomes 0 after i restrictions, for some $i < R$. When $\sum_{j=1}^{T} \Phi(G_j, \rho_1, \ldots, \rho_i) = 0$, with probability at least $1 - \sum_{j=1}^{T} p_{G_j}$, each of the sub-formulas is tiny. At this stage, the oracle gate at the top depends on at most $O(T \cdot k)$ many variables and its potential is set to T^2k^3 now.[20] Thus, when $\Phi(G, \rho_1, \ldots, \rho_R)$ becomes 0 following a further set of restrictions, the probability the oracle depends on at most $\log N$ variables is at most $1/N^4$ using an argument similar to that of case II (the pure oracle case). Setting $p_G = \sum_{i=1}^{T} p_{G_i} + 1/N^4$, we see that property 2 holds.

Case IV: G is a Formula with Oracle Leaves

For this case, we treat G as an oracle formula whose leaves are either literals or oracle gates, ignoring the sub-formulas rooted at the oracle gates. Intuitively, this case occurs when the potential function for the maximal sub-formulas associated with the oracle leaves have already been constructed and we need to use them as inductive blocks to build $\Phi(G)$. Observe that Case I is a special case of this.

Suppose G is an oracle formula of size S, with m oracle leaves $O_1, \ldots, O_m$. Let $\Phi(O_i)$ be the potential of the sub-formula corresponding to O_i. Also, let c_O be the maximum of the c_{O_i}s of the sub-formulas associated with the oracle leaves.

At the end of each pseudorandom restriction, whenever an oracle formula with a top oracle gate depends on at most $b = 10$ variables, we transform the oracle gate into a formula of size 2^b. This ensures easier dealing of oracle gates when they become very small, since they have now become constant-sized formulas. Now, the difficulty in analysis arises when we have many oracle leaves which have become tiny (depend on at most $\log N$ variables), but still have not collapsed to b variables. Such oracle formulas have their potentials down to 0, and yet in total, could be supported on many variables. We explicitly keep track of such oracles. Define an oracle leaf as *active* if it has not been

[20] Analogous to Case II (the pure oracle case), a potential of $O(T^2k^2)$ would be sufficient here, as this would mean the probability of leaving a variable unrestricted by the composition of all the restrictions is at most $1/Tk$. We use T^2k^3 for notational convenience (to eliminate the constant in the potential).

transformed into a formula of size at most 2^b yet. Let m_{tiny} be the number of active tiny oracles, i.e. oracle leaves which have become tiny but have not been transformed to a 2^b-sized formula.

As before, we proceed with the process of applying restrictions iteratively. At the beginning, the potential of G is set as:

$$\Phi(G) = \begin{cases} S + m_{\text{tiny}} \cdot k^2 + \left(\sum_{j=1}^{m} \Phi(O_j)\right) \cdot k^4 & \text{if } S > 80k, \\ m_{\text{tiny}} \cdot k^2 + \left(\sum_{j=1}^{m} \Phi(O_j)\right) \cdot k^4 & \text{otherwise.} \end{cases}$$

To show shrinkage under expectation of $\Phi(G)$ under $\rho \sim \mathcal{D}$, we first consider the formula itself and suppose that $S > 80k$. The number of active oracles (oracles which have not collapsed to a constant-sized formula yet) are at most $(m - m_{\text{tiny}}) + m_{\text{tiny}}$. Using Lemma 4.62, we see that the formula of size S can be split into $6S/k$ sub-formulas of size at most k, where each formula is connected to at most two other formulas in the collection. Now, there are at least $6S/k - m$ sub-formulas of size at most k which do not contain an active oracle leaf. Thus, the total size of all these formulas under ρ is at most $O(S/k)$ from Lemma 4.63. For those sub-formulas which contain an active oracle leaf, the total size is at most $m \cdot O(k)$ after ρ (the constant term accounts for when an active oracle leaf collapses to a formula over at most b variables).

Next, we consider the active tiny oracles. Observe that the probability that an active tiny oracle, which depends on at most $\log N$ variables, ends up having larger than b variables under ρ is at most

$$\binom{\log N}{b} \cdot k^{-b/2} \leq (e \log N/b)^b \cdot k^{-b/2} \leq (1/\log^5 N) \leq 1/k$$

Thus, after the application of ρ, the number of active tiny oracles becomes at most $\frac{m_{\text{tiny}}}{k} + (m - m_{\text{tiny}})$ in expectation. Finally, by the induction over the maximal sub-formulas, every $\Phi(O_j)$ reduces by a factor of c_O/k in expectation after ρ.

Using the fact that $\sum_{j=1}^{m} \Phi(O_j) \geq m - m_{\text{tiny}}$, the expected potential of G under ρ is at most

$$\begin{aligned} &\leq c_1 \cdot \frac{S}{k} + c_2 \cdot k \left(\sum_{j=1}^{m} \Phi(O_j) + m_{\text{tiny}}\right) + \left(\frac{m_{\text{tiny}}}{k} + \sum_{j=1}^{m} \Phi(O_j)\right) \cdot k^2 \\ &\qquad + \left(\sum_{j=1}^{m} \Phi(O_j)\right) \cdot \frac{c_O}{k} \cdot k^4 \\ &\leq c_1 \cdot \frac{S}{k} + m_{\text{tiny}} \cdot k^2 \cdot \frac{c_2 + 1}{k} + \left(\sum_{j=1}^{m} \Phi(O_j)\right) \cdot k^4 \cdot \frac{c_O + c_2/k^2 + 1/k}{k} \end{aligned}$$

Setting $c_G = \max(c_1, c_2 + 1, c_O + c_2/k^2 + 1/k)$, we show that Φ reduces by a factor c_G/k in expectation.

For the second property, observe that $\Phi(G)$ becomes 0 only after $S \leq 80k$ at some point, in addition to m_{tiny} and $\sum_{j=1}^{m} \Phi(O_j)$ becoming 0. In particular, this implies that the oracles have further collapsed to formulas over b variables. Since, each transformed oracle adds at most 2^b leaves, when $\Phi(G)$ becomes 0, the size of the whole formula is at most $2^b \cdot 80k$ and thus, it is tiny by setting $c_{\text{tiny}} = 2^b \cdot 80$. We set $p_G = \sum_{i=1}^{m} p_{O_i}$.

We are now ready to prove the MCSP lower bound. For some constant c which will be fixed later, suppose that there exists $\varepsilon > 0$, oracle $\mathcal{O}$, and a sequence of $\mathcal{O}$-oracle formulas $\{F_N\}$ that computes $\mathsf{MCSP}[n^c, 2^{\varepsilon n/3}]$, such that $\mathsf{Formula}_3(F_N) \leq N^{2-\varepsilon}$ and its adaptivity is $a(N) = o(\log N / \log\log N)$. The value of $p_F \leq N^2 \cdot N^{-4} \leq 1/N^2$.

Furthermore, we see that the maximum value of c_F is obtained when Case IV inductively repeats itself at each layer of adaptivity, i.e. for each oracle gate, at least one of its inputs is fed by a maximal sub-formula which satisfies the structure analysed in Case IV. Thus, c_F can be upper bounded as

$$c_F \leq \max\{c_1, c_2 + 1, 72\} + a(N)(c_2/k^2 + 1/k) \leq \max\{c_1, c_2 + 1, 72\} + o(1)$$

which is a constant. Note that the constant 72 comes from Case I.

Next, observe that $\Phi(F) \leq \mathsf{Formula}_3(F) \cdot k^{O(a(N))} \leq N^{2-\varepsilon+o(1)}$. Intuitively, the inductive definition of Φ for a formula F with a fan-in T oracle gate at the top (Case III), is the measure of each input sub-formula repeated exactly T^2 times (ignoring the $\mathsf{poly}(k)$ term). In other words, Φ can be viewed as the size of the De Morgan formula obtained by transforming an oracle into an equivalent formula of size T^3, where each of the T inputs occurs exactly T^2 times. Furthermore, since $a(N) = o(\log N / \log\log N)$, $k^{O(a(N))}$ is constrained to be at most $N^{o(1)}$.

From property 1, $\Phi(F)$ drops by a factor k/c_F in expectation, after each restriction from $\mathcal{D}$. Thus, after $r = \lceil 2\log_{k/c_F} N \rceil + 3$ many restrictions, the expected value of $\Phi(F, \rho_1, \ldots, \rho_r)$ is less than $1/20$. Applying Markov's inequality, with probability at least $0.95 - p_F \geq 0.9$, there exists a restriction ρ (which is a composition of r independent restrictions) such that $F|_\rho$ depends on at most $O(k)$ many variables.

Furthermore, the composition of r independent restrictions ρ keeps a variable unrestricted with probability at least $k^{-r/2}$ and since $k \geq 2$, ρ is pairwise independent. Expanding this using the definition of r, we see that $k^{-r/2} \geq ((c_F)^r \cdot k^3 \cdot \Phi(F))^{-1/2} \geq N^{(2-\varepsilon+o(1))(-1/2)} \geq 1/N^{1-\varepsilon/2+o(1)}$. This equation holds because $r = O(\log N / \log\log N)$ and thus, $(c_F)^r = N^{o(1)}$. Using Chebyshev's inequality (Lemma 4.6), we see that with probability at least 0.9, $|\rho^{-1}(*)| \geq N^{\varepsilon/2-o(1)}$. Put together, we see that there exists a

fixed restriction ρ such that $F|_\rho$ is at most $O(k)$ and ρ leaves at least $N^{\varepsilon/2-o(1)}$ variables unrestricted.

The contradiction is proved in a similar manner to the lower bounds seen earlier. We construct $w^Y = \rho \circ 0^N$ which is a YES instance, because from Lemma 4.8 we see that the circuit complexity of w^Y is at most $\mathsf{poly}(\log N) \cdot r \leq n^c$, for some $c \geq 1$ which is fixed to be large enough. Let U be the variables on which $F|_\rho$ depends on. After setting the variables in U to 0, there exists at least $2^{N^{\varepsilon/2-o(1)}-O(\mathsf{poly}(\log N))} > 2^{O(N^{\varepsilon/3}\log N)}$ many ways of assigning variables in $V = \rho^{-1}(*) \setminus U$, where the latter quantity is the number of circuits on n inputs of size at most $N^{\varepsilon/3}$. Thus, there exists an assignment h to the variables in V, such that any N-length truth table which agrees with h has circuit complexity at least $N^{\varepsilon/3}$. In particular, we extend ρ to get a NO instance w^N by fixing U to 0 and the remaining variables in $\rho^{-1}(*)$ to h. Since $F|_\rho$ only depends on U, it accepts both w^Y and w^N, which contradicts the fact that F computes $\mathsf{MCSP}[n^c, 2^{\varepsilon n/3}]$. □

4.9 Concluding Remarks and Open Problems

Hardness magnification shows that obtaining a refined understanding of weak computational models is an approach to major complexity lower bounds, such as separating EXP from NC^1. As discussed in Sections 3.1.2 and 4.1.1, its different instantiations are connected to a few basic questions in Complexity Theory, including the power of non-monotone operations, learnability of circuit classes, and pseudorandomness.

One of the main conceptual contributions of this chapter is to identify a challenge when implementing this strategy for lower bounds. Quoting the influential article [RR97] that introduced the natural proofs barrier,

> *"We do not conclude that researchers should give up on proving serious lower bounds. Quite the contrary, by classifying a large number of techniques that are unable to do the job we hope to focus research in a more fruitful direction."*
>
> Razborov and Rudich [RR97, Section 6]

We share a similar opinion with respect to hardness magnification and the barrier identified in this chapter. While locality provides a unified explanation for the difficulty of adapting combinatorial lower bound techniques to exploit most (if not all) known magnification frontiers, it might be possible to discover new HM frontiers whose associated lower bound techniques in Item 3 are sensitive to the presence of small fan-in oracles.

[Oli19] provides one such instance, by introducing a randomised analogue of Kt complexity called MrKtP and showing that it exhibits magnification phenomena similar to MKtP. In particular, if MrKtP has a super-linear lower bound against Formula-XOR, then

$\mathsf{Promise\text{-}BPE} \not\subseteq \mathsf{NC}^1$. The proof does provide a circuit construction for MrKtP with oracle gates of small fan-in. Nevertheless, an unconditional *uniform complexity lower bound* for MrKtP in the same paper, via an indirect diagonalisation argument that explores the theory of pseudorandomness, does not extend to algorithms with small fan-in oracles. Very simply put, this is because the easiness of MrKtP does not help us much when trying to diagonalise against computations with small-fan in oracles, as the problem refers only to *standard computations*. This shows the existence of lower bound techniques and computational problems that, together, can be sensitive to local oracles.

It would also be useful to investigate the locality of additional lower bound techniques. Can we, for example, come up with non-localisable lower bounds similar to Theorem 4.48 which would be above the magnification threshold and work for a problem more closely related to the one from the corresponding HM frontier?

Alternatively, it might be possible to establish magnification theorems using a technique that does not produce circuits with small fan-in oracles. Furthermore, recent works suggest approaches such as the *Explicit Obstructions* framework by [CJW20], or the *meta-computational view of PRG constructions* by [Hir20b], as potential ways of bypassing the locality barrier.

Even if one is pessimistic about these possibilities, we believe that an important contribution of the theory of hardness magnification is to break the divide between "weak" and "strong" circuit classes advocated by the natural proofs barrier. Magnification indicates that investigating specific computational problems can actually make a big difference, and perhaps the computational problem for which the lower bound is being shown cannot be left out of the picture. For example, while strong lower bounds are known for problems like $\mathsf{Majority}$ against a weak class like $\mathsf{AC}^0[2]$, showing even *weak lower bounds* for a seemingly harder problem like $\mathsf{MKtP}[(\log N)^d]$ against a weaker class like $\mathsf{AC}^0\text{-}\mathsf{XOR}$ has seemed out of reach so far.

In this sense, we believe that the theory of magnification has allowed us take a second look at combinatorial lower bound methods and natural proofs, and try to understand why showing lower bounds for certain problems is harder than others, and that it deserves further investigation.

Chapter 5

The Structure of Learnability beyond P/poly

5.1 Introduction

What is the complexity of learning polynomial-size circuits? Despite extensive research on this question, our knowledge is still fairly sparse. For weak concept classes such as decision trees [LMN93, KM93], DNFs [LMN93, Jac97] or even constant-depth circuits with parity gates [CIKK16], reasonably efficient learning algorithms under the uniform distribution are known for various models of learning. For stronger concept classes, learning is believed to be hard, but the evidence for this is not as strong as one might hope. Cryptographic assumptions such as the existence of one-way functions are known to imply that learning polynomial-size circuits is hard [KV94a, GGM86]. However, we still seem far from showing that PAC-learning polynomial-size circuits is NP-hard - indeed [ABX08] give negative results for certain kinds of black-box reductions to learning.

In this chapter, we adopt a fresh perspective of approaching the learnability question from above, i.e. via circuit classes which are more powerful than P/poly. We consider commonly held beliefs about the complexity of learning, and establish these beliefs unconditionally for strong concept classes such as PSPACE/poly and EXP/poly. Of course the very learnability of these concept classes has some unlikely implications, e.g. that these classes are approximable by efficient Boolean circuits. The point is that this is still consistent with our complexity-theoretic understanding, and we would like to know what current techniques are capable of proving *unconditionally* about learning. Partly this is to understand the limitations of current techniques, and partly this is to understand what structural properties of the stronger concept classes enable us to show unconditional results about them.

We begin by outlining our main results and comparing them with previous work.

5.1.1 Unconditional results for hardness of learning

Our first set of results deals with unconditional hardness of learning circuit classes. Most complexity theorists believe that learning polynomial-size circuits is unconditionally hard, but of course proving this is at least as hard as the P vs NP problem. We ask: what is the smallest concept class $\mathcal{C}/\mathsf{poly}$[1] for which we can *prove* learning to be hard? Clearly, if we can prove that $\mathcal{C}/\mathsf{poly}$ cannot be approximated by efficient circuits, i.e. there does not even *exist* a good hypothesis for all concepts in the class, then hardness of learning follows. This observation implies for example that learning MAEXP is hard, by using known circuit lower bounds for this class [BFT98].

But can we show hardness of learning unconditionally for some concept class where it is consistent with our current understanding of complexity theory that a good hypothesis exists for every concept in the class? We give an affirmative answer by ruling out PAC-learning with membership and equivalence queries unconditionally for the class $\mathsf{BPE}/\mathsf{poly}$.

The notion of PAC-learning $\mathcal{C}/\mathsf{poly}$, for a uniform class $\mathcal{C}$ above P such as EXP or BPE, can have different interpretations. In this chapter we use the definition of efficiently PAC-learning a class $\mathcal{C}/\mathsf{poly}$, as one which runs in $\mathsf{poly}(n)$ time using P/poly as its hypothesis class (assume that the accuracy ε and confidence δ are both $1/\mathsf{poly}(n)$).

To motivate this, note that standard definitions for PAC-learning (cf. [KV94b]) consider the task of learning to be efficient if it is polynomial in the size of the target concept over n inputs (where ε and δ are both $1/\mathsf{poly}(n)$) and the hypothesis class is P/poly.[2] Following this, we can consider efficient PAC-learning for weaker classes (like P/poly) as learning in $\mathsf{poly}(n)$ time using P/poly-hypotheses, and see that this naturally extends to the concept class $\mathcal{C}/\mathsf{poly}$ as the size of the target concept is *still* polynomial in the input size n. Alternatively, one can also consider polynomial time learners where the hypothesis has a polynomial size description, but should be evaluated with respect to an oracle to some function in $\mathcal{C}$. The learning algorithms could also have access to more resources, for example have access to some $\mathcal{C}$-oracle or even have the ability to run in $2^n/n^{\omega(1)}$ time. For the classes $\mathcal{C}$ we consider, PAC-learnability of $\mathcal{C}/\mathsf{poly}$ in $\mathsf{poly}(n)$ time using P/poly-hypotheses is still consistent with our current understanding of complexity theory (as we do not have any unconditional average-case lower bounds for $\mathcal{C}$ against P/poly), and as such, we use this interpretation of learning $\mathcal{C}/\mathsf{poly}$ for proving unconditional results.

[1] For any uniform complexity class $\mathcal{C}$, define the class $\mathcal{C}/\mathsf{poly}$ as the set of languages L for which there is a $\mathcal{C}$-machine M and a family of strings $\{a_n\}$, where $a_n \in \{0,1\}^{\mathsf{poly}(n)}$, such that for every $x \in \{0,1\}^n$, $x \in L \iff M$ accepts (x, a_n)

[2] In general, the definition requires the hypothesis class H to be *polynomially evaluatable*, which means that there exists an algorithm that on input any instance $x \in \{0,1\}^n$ and an encoding of the hypothesis $h \in H_n$, outputs the value $h(x)$ in time polynomial in n and the size of the hypothesis encoding. It is well known that P/poly is polynomially evaluatable.

$\mathsf{BPE/poly}$ can equivalently be defined as the class of languages computable by polynomial size circuit families with oracle gates to some function in BPE, with the oracle query size restricted to $O(n)$. We prove the unconditional hardness of learning $\mathsf{BPE/poly}$ in polynomial time using membership queries even over the uniform distribution using $\mathsf{P/poly}$ as the hypothesis class. Hardness of exactly learning $\mathsf{BPE/poly}$ with membership and equivalence queries, even using randomised algorithms follows directly from this via [Ang88].

Theorem 5.1. *For every constant $k \in \mathbb{N}$, $\mathsf{BPE/poly}$ cannot be $(1/2 - 1/n^k, 1/n)$-learnt over the uniform distribution using membership queries by randomised learning algorithms running in polynomial time.*

To prove this, we adapt techniques used by [KKO13, OS17] to show that randomised PAC-learning algorithms imply circuit lower bounds. [TV07] show the existence of a PSPACE-Complete function f^* which is in $\mathsf{DSPACE}[n]$, such that f^* is downward self-reducible and self-correctible (see Section 5.2 for definitions). Using the techniques of [KKO13], along with the fact that f^* belongs to BPE, we see that PSPACE collapses to BPP. Using a padding argument and diagonalising $\mathsf{DSPACE}[2^{O(n)}]$ against functions which can be approximated by polynomial-sized circuits, we obtain a contradiction to the fact that for every function in $\mathsf{BPE/poly}$, the learner gives a hypothesis circuit which approximates it well.

5.1.2 Robustness of learning

We believe that polynomial-size circuits are hard to learn in a *robust* sense, i.e. that the precise details of the learning model do not matter. Hardness should hold irrespective of whether we consider PAC-learning or learning over the uniform distribution, worst-case learning or average-case learning over some samplable distribution on concepts, and whether or not the learning model is allowed to use membership queries. We do not know how to show that this robustness holds for $\mathsf{P/poly}$, but we are able to show it unconditionally for $\mathsf{EXP/poly}$ and $\mathsf{PSPACE/poly}$.

Consider the class $\mathsf{EXP/poly}$, which can be equivalently defined as the circuit class $\mathsf{P}^{\mathsf{EXP}}\mathsf{/poly}$, i.e. the class of languages that can be computed by a polynomial sized circuit family with EXP oracle gates.

Showing non-trivial derandomisation of BPP, i.e. $\mathsf{EXP} \neq \mathsf{BPP}$, is one of the most fundamental questions in complexity theory.[3] We prove that the problem of non-trivial derandomisation of BPP is equivalent to the hardness of learning $\mathsf{EXP/poly}$ efficiently

[3]It is worth mentioning that [IW01] show that $\mathsf{EXP} \neq \mathsf{BPP}$ is equivalent to the fact that BPP can be derandomised on average in deterministic sub-exponential time (over infinitely many input lengths).

in most standard models of PAC-learning. In addition, these results extend to not just showing that EXP/poly is hard to learn in the worst-case, but also on average with respect to quasi-polynomially samplable distributions over EXP/poly (or polynomially samplable distributions depending on the function representation output by the sampler). This also gives us an intriguing situation, where hardness of learning EXP/poly using random examples also implies the hardness of learning EXP/poly using membership queries.

The following results are stated for hardness of strong learning. However, they also hold for the setting of weak learnability, by standard equivalences between weak learning and strong learning for PAC-learners [FS97].

Theorem 5.2 (**Equivalences for hardness of learning** EXP/poly)**.** *The following statements are equivalent.*

(a) ***Non-trivial derandomisation of BPP****:* $\mathsf{EXP} \neq \mathsf{BPP}$*.*

(b) ***Hardness of PAC-learning*** EXP/poly ***in the worst-case using random examples*** *: For every large enough constant c,* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time using random examples.*

(c) ***Hardness of PAC-learning*** EXP/poly ***in the worst-case using membership queries****: For every large enough constant c,* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time using membership queries.*

(d) ***Hardness of PAC-learning*** EXP/poly ***on average using random examples****: For every large enough constant c,* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time on average using random examples, with respect to polynomially samplable distributions over* EXP/poly*.*

(e) ***Hardness of PAC-learning*** EXP/poly ***on average using membership queries****: For every large enough constant c,* EXP/poly *is not $(1/n^c, 1/20n)$-PAC-learnable in polynomial time on average using membership queries, with respect to polynomially samplable distributions over* EXP/poly*.*

A contrasting result to this is the equivalence between the existence of one-way functions (OWFs) and the hardness of learning P/poly in polynomial time on average using random examples, with respect to polynomially samplable distributions over P/poly [IL90, BFKL93]. Theorem 5.2 not only lends an analogous equivalence between a complexity theoretic assumption that BPP has a non-trivial derandomisation and the hardness of learning EXP/poly in polynomial time on average using random examples, but also extends this equivalence to hardness of learning EXP/poly efficiently in the worst-case.

Note that such an equivalence between the existence of OWFs and hardness of learning $\mathsf{P/poly}$ efficiently in the worst-case has been open for decades.[4]

Furthermore, our proof techniques also let us extend all these equivalences to the case where $\mathcal{C} = \mathsf{PSPACE}$.

Corollary 5.3. *The following statements are equivalent.*

1. $\mathsf{PSPACE} \neq \mathsf{BPP}$.
2. *For every large enough constant c,* $\mathsf{PSPACE/poly}$ *is not* $(1/n^c, 1/20n)$*-PAC-learnable in polynomial time using random examples (also using membership queries).*
3. *For every large enough constant c,* $\mathsf{PSPACE/poly}$ *is not* $(1/n^c, 1/20n)$*-PAC-learnable in polynomial time on average using random examples (also using membership queries), with respect to polynomially samplable distributions over* $\mathsf{PSPACE/poly}$.

Essentially, the proof of showing conditional hardness of PAC-learning $\mathsf{EXP/poly}$ uses the fact that strongly learning $\mathsf{EXP/poly}$ using random examples over the *uniform distribution* implies that $\mathsf{EXP} = \mathsf{BPP}$. This also means that the hardest distribution to learn $\mathsf{EXP/poly}$ is over the uniform distribution. The same ideas hold for PAC-learning $\mathsf{EXP/poly}$ using membership queries too.

Our techniques used to show these equivalences are inspired from results on uniform derandomisation by [IW01, TV07], which were further used by [FK09, KKO13] to show circuit lower bounds based on the existence of learning algorithms. We use special properties of functions in EXP and PSPACE like downward self-reducibility and self-correctibility to show that learning these functions would imply a collapse for EXP or PSPACE to BPP.

5.1.3 Reducing Succinct Search to Decision for Learning

Recently, [CIKK16] established an important connection between *natural proofs* and *learning*. They showed that natural proofs of strong lower bounds against a circuit class $\mathcal{C}/\mathsf{poly}$ imply efficient learning algorithms for $\mathcal{C}/\mathsf{poly}$ over the uniform distribution with membership queries, as long as the class $\mathcal{C}/\mathsf{poly}$ satisfies some mild closure properties. One way to interpret their result is as an *approximate search to decision* reduction for learning. The decision version of learning polynomial-size circuits is the language MCSP consisting of truth tables of functions that have small circuits, i.e. for which a good hypothesis exists. The search version is to find a small circuit for a positive instance of MCSP. [CIKK16] show that if MCSP is polynomial-time decidable (which implies the existence of natural proofs against $\mathsf{P/poly}$), then the search version of MCSP can be solved

[4] In particular, we do not know if hardness of learning $\mathsf{P/poly}$ efficiently using random examples in the worst-case implies OWFs.

approximately, in the sense that we can efficiently compute a polynomially larger sized circuit that approximates the truth table well.

The language $\mathsf{R_{Kt}}$ (resp. $\mathsf{R_{KS}}$) of strings with high Kt complexity (resp. high KS complexity) plays an analogous role to MCSP in the theory of learning $\mathsf{EXP/poly}$ (resp. $\mathsf{PSPACE/poly}$). We ask if search to decision reductions can be established for these languages as well. However, it is unclear a priori what it would mean to solve search efficiently for a problem that does not have polynomial-size proofs or witnesses. We introduce the notion of *succinct search*. To efficiently solve a search problem succinctly is to efficiently compute for any YES instance of the problem, a circuit that encodes a possibly exponential-size proof for the instance. We use the PCP theorem for NEXP [BFL91] and the Easy Witness Lemma [IKW02] to show that for the classes $\mathsf{PSPACE}, \mathsf{EXP}$ and NEXP, efficient decidability of the class is equivalent to efficiently solving succinct search for every language in the class. We then use results from [ABK+06] to argue that for $\mathsf{R_{Kt}}$ and $\mathsf{R_{KS}}$, efficient solvability is equivalent to solving succinct search efficiently. Note that this connection is for succinctly solving the search problem *exactly* rather than just for approximate search as in [CIKK16].

Theorem 5.4 (**Equivalence of Succinct Search and Decision for Learning $\mathsf{EXP/poly}$ and $\mathsf{PSPACE/poly}$**). *Let L be $\mathsf{R_{Kt}}$ or $\mathsf{R_{KS}}$. $L \in \mathsf{BPP}$ iff for each polynomial time verifier V for L, succinct search is efficiently solvable for L with respect to V.*

5.1.4 Barriers for Establishing NP-Hardness of Learning.

We next look at questions pertaining to hardness of learning classes of the form $\mathcal{C}/\mathsf{poly}$, where $\mathcal{C} \subseteq \mathsf{PH}$. We focus on the hardness of PAC-learning $\mathcal{C}/\mathsf{poly}$ with random examples. In this section, we consider the limitations of proving the NP-Hardness of PAC-learning $\mathsf{NP/poly}$, i.e. the class of polynomial size non-deterministic circuits, using random examples, via a black-box reduction from deciding SAT.

Informally, a black-box reduction from problem A to B, solves A given access to any oracle solving B. Black-box reductions have been ubiquitously used in complexity theory to prove conditional lower bounds. However, for many fundamental questions in complexity theory, there have been results showing why such reductions are limited in power. Various works have conditionally ruled out special-cases of black-box reductions for showing average-case hardness of NP [FF93, BT06], existence of one-way functions [AGGM06, ABX08, BB15] and the existence of hitting set generators [HW19], from hardness of SAT.

For the case of showing hardness of learnability, a B-adaptive black-box reduction R from some language L to PAC-learning a class $\mathcal{C}$ using random examples is defined by two phases

- The first phase consists of B adaptive rounds of probabilistic polynomial time algorithms, each of which generates queries to the learner oracle. Each round uses the input z to the reduction, fresh randomness and the hypotheses returned by the $\mathcal{C}$-learner oracle in the previous rounds, and constructs joint distributions (that serve as example oracles for the learner). It then samples labeled examples independently from them as queries to the learner oracle.

- In the second phase, a probabilistic polynomial time algorithm takes all the hypotheses from the first phase and decides whether $z \in L$, with high probability.

[ABX08] study the question of the existence of black-box Turing reductions from any language in NP to PAC-learning P/poly using random examples. They consider a strongly black-box reduction, where a reduction is strongly black-box if it runs correctly given any oracle for the learner, as well as the hypotheses output by the learner and for a special case of such a reduction, where the access to the learner and the hypothesis oracles is additionally non-adaptive, show that such a reduction from SAT to PAC-learning P/poly using random examples collapses NP to CoAM (which implies a collapse of PH to the second level). Additionally, they show that if any language L reduces to PAC-learning P/poly using random examples via an $O(1)$-adaptive black-box reduction, then the hardness of L implies the existence of an auxiliary-input one-way function (which is a major breakthrough in cryptography if $L = \mathsf{SAT}$).[5]

We define a natural special-case of such a reduction, called an oblivious strongly black-box reduction, where the obliviousness of a reduction implies that the queries made to the learner do not depend on the input z to the reduction and try to understand its limitations for showing NP-hardness of PAC-learning NP/poly. At a first glance, ruling out oblivious reductions may seem very restrictive, since ideally, one would like to allow reductions whose queries to the learner can depend on the input to the reduction. However, we observe the proof of Corollary 5.3 which shows hardness of PAC-learning PSPACE/poly assuming PSPACE $\neq$ BPP and reformulate it as an oblivious black-box reduction of the form defined above. In particular, for f^* being the PSPACE-Complete function given by [TV07] which is downward self-reducible and self-correctible, we observe that

Lemma 5.5. *There exists an oblivious, n-adaptive, strongly black-box reduction from deciding f^* to learning* PSPACE/poly *using random examples over the uniform distribution.*

On the other hand, for the case of learning NP/poly using random examples, we show that oblivious strongly black-box reductions from SAT imply a collapse of the polynomial hierarchy. Our main result for the section is

[5] They also show the impossibility of Karp reductions from SAT to PAC-learning P/poly using random examples, unless NP collapses to SZKA.

Theorem 5.6 (Informal)**.** *If there exists an oblivious,* $\mathsf{poly}(n)$*-adaptive, strongly black-box reduction from deciding* SAT *to learning* $\mathsf{NP/poly}$ *using random examples over polynomially samplable distributions, then* PH *collapses to the third level.*[6]

Theorem 5.6 implies that standard techniques that are useful for worst-case to average-case reductions, pseudo-random generator constructions from uniform hardness assumptions and in particular, hardness of efficiently PAC-learning classes like $\mathsf{PSPACE/poly}$, cannot be used to show the NP-hardness of PAC-learning $\mathsf{NP/poly}$ in polynomial time using random examples.

Theorem 5.6 compares to some previous results in the following way:

- It shows a conditional impossibility result by ruling out a restricted version of adaptive, strongly black-box reductions to learning $\mathsf{P/poly}$ using random examples, in contrast to [ABX08], who only rule out fully non-adaptive, strongly black-box reductions, from a slightly weaker assumption ($\mathsf{NP} \not\subseteq \mathsf{CoAM}$).

- Furthermore, the result by [HW19] which conditionally rules out a non-adaptive black box reduction from deciding SAT to breaking a Hitting Set Generator (HSG), in turn rules out fully non-adaptive, strongly black-box reductions from SAT to learning $\mathsf{NP/poly}$ using membership queries over the uniform distribution (by suitably changing the definition of the reduction to the learner).

  Indeed, the ideas of [IW01] can be used to show that hardness of learning $\mathsf{NP/poly}$ using membership queries over the uniform distribution, implies the existence of a hitting set generator which hits sufficiently dense circuits. We strengthen this observation by not only extending the reduction to a restricted version of the adaptive case, but also by ruling out a weaker reduction to learning $\mathsf{NP/poly}$ with random examples.

- In a similar way, [GV08] conditionally rule out the existence of mildly adaptive (each query length up to n, where n is the length of the input instance, appears in very few levels of adaptivity), strongly black-box reductions from an EXP-Complete problem to learning $\mathsf{NP/poly}$ using membership queries (and in fact, learning $\mathsf{EXP/poly}$).

  Our result rules out the restricted cases of mildly adaptive, strongly black-box reductions which show the NP-hardness of learning $\mathsf{NP/poly}$ using random examples and hence, is a conceptual strengthening of [GV08], as we rule out a hardness result from a stronger assumption.

[6] We actually show a stronger result that the existence of such a reduction implies that $\mathsf{NP} \subseteq \mathsf{CoAM}^{\mathsf{poly}}$, where $\mathsf{CoAM}^{\mathsf{poly}}$ is the class of languages recognised by constant-round CoAM protocols with advice, where we require proper acceptance/rejection probabilities only when the advice is correct.

It is worth noting that our result has no implications for showing the impossibility of adaptive, black-box NP-Hardness reductions which imply the average-case hardness for NP [FF93, BT06], existence of one-way functions [AGGM06, ABX08] or the existence of HSGs [GV08, HW19].

Overview of the proof: The proof of Theorem 5.6 builds on the Feigenbaum-Fortnow [FF93] protocol, which simulates a type of non-adaptive randomised reduction A from SAT to an NP problem $\mathcal{Q}$, by an AM protocol with polynomial-sized advice, and shows that coNP $\subseteq$ NP/poly.[7]

Suppose that on input x, A makes q non-adaptive queries to $\mathcal{Q}$, sampled independently from certain distribution X. Very briefly, their AM protocol does the following. For K large enough, the verifier first generates K tuples of q non-adaptive queries by running $A(x)$ independently K times. The verifier asks the prover to send a witness to each query which is a YES instance (which it can verify easily). This ensures that the prover cannot cheat if the query is a NO instance and the only way it can cheat is by claiming a YES instance to be a NO instance. Now, if the verifier has the proportion p of YES instances of $\mathcal{Q}$ over the distribution X, then with high probability it knows that the number of YES instances among the Kq queries is concentrated around $q \cdot (pK \pm O(\sqrt{K}))$. The verifier answers with a reject if the number of YES instances is much lesser than pqK.

The honest prover answers each query correctly (with correct witnesses if necessary) and with high probability, the number of YES instances are close to the expectation. Hence, the verifier can pick any of K runs of $A(x)$ using the prover's answers to its queries and the output will be correct with high probability. On the other hand, with high probability, the cheating prover cannot cheat on more than $O(q\sqrt{K})$ YES instances, with high probability. If we choose $K >> O(q\sqrt{K})$, then on most of the K independent runs of A, all its queries are answered correctly and the reduction gives the correct answer. Thus, if we pick one of the runs at random and get $A(x)$ by using the prover's answers to its queries, the verifier answers wrongly with low probability.

Consider an oblivious, B-adaptive, strongly black-box reduction R from L to an oracle which learns NP/poly. Suppose we are able to fix $S_1, \ldots, S_t$, which are sets of labeled examples drawn independently from the joint distributions $(X_1, f_1(X_1)), \ldots, (X_t, f_t(X_t))$, such that $f_1, \ldots, f_t \in$ NP/poly, and are the queries made to the learner. Furthermore, let $h_1, \ldots, h_t$ be a set of fixed hypotheses circuits, some of which are used to generate $S_1, \ldots, S_t$, such that each h_i $(1 - \varepsilon_0)$-approximates f_i over X_i, for some $\varepsilon_0 > 0$. Because R is strongly black-box, each hypothesis is also accessed as an oracle and we see that L is decided by the algorithm M in the second phase, which has access to $h_1, \ldots, h_t$.

[7] Their motivation (and [BT06]) was to rule out certain kinds of non-adaptive, worst-case to average-case black-box reductions for NP.

Now, the t oracles to M can be replaced by a single oracle $\mathcal{O}$ which takes as input $i \in [t]$ and $y \in \{0,1\}^n$, and outputs $h_i(y)$ ($\mathcal{O}$ can be thought of as a table with t rows and 2^n columns). We then adapt the techniques of [FF93] to design an AM protocol for L with polynomial sized advice, where the verifier expects that the prover answers according to $\mathcal{O}$.

The obliviousness of the reduction helps us in fixing the queries made by R, and implicitly, the corresponding hypotheses output by the oracle. In other words, this helps us fix the proportions of YES instances for each f_i non-uniformly, as the queries generated to the learner do not depend on the input to the reduction. We do this by inductively fixing the queries made by the reduction starting from the first round of adaptivity. Fixing a "good" polynomial-sized random string r^* used by the first phase non-uniformly (using Adleman's trick), we first get the queries to the learner made in the first round.

For any other round $b \geq 2$, assume that the queries to the learner up to round $(b-1)$ and the functionality of the hypothesis oracles used to generate them up to round $(b-2)$ are fixed. Using the fact that r^* is also fixed, we can consider the set of all tuples of joint distributions that can be generated in the b^{th} round depending on the answers to the oracle queries of the hypotheses seen so far, and arbitrarily choose one of them. Note that, this implicitly fixes the functionality of the hypothesis oracles for the queries generated in round $b-1$. We continue this process and fix all the queries made to the learner in all the rounds from the first phase.

5.2 Preliminaries

5.2.1 Samplability and Learnability

Let $\mathcal{C} = \{\mathcal{C}_n\}$, where $\mathcal{C}_n \subseteq \mathcal{F}_n$ is a class of functions over $\{0,1\}^n$ and $\mathcal{D} = \{\mathcal{D}_n\}$ be a distribution family over $\{0,1\}^*$, where $\mathcal{D}_n$ is a distribution over $\{0,1\}^n$.

For definitions on worst-case learnability we refer the reader to Section 2.2. To define learnability on average, let $\mathcal{P} = \{\mathcal{P}_n\}$ be a distribution ensemble over $\mathcal{C}$, where $\mathcal{P}_n$ is a distribution over $\mathcal{C}_n$.

Definition 5.7 (**Samplable distributions**). *Let $\mathcal{P}$ be a distribution ensemble over $\mathcal{C}$. For $N = 2^n$ and any non-decreasing function $S(N) \geq N$, we say that $\mathcal{P}$ is samplable in time $S(N)$, if there exists a randomised algorithm A that uses $m(N)$ bits of randomness (where $m(N) \leq S(N)$) to output a string of length N, such that $A(1^N, y)$ is distributed identically to $\mathcal{P}_n$ over the random choice of $y \in \{0,1\}^{m(N)}$, and A runs in time $S(N)$.*

In other words, if y is picked uniformly at random from $\{0,1\}^{m(N)}$ then the output of $A(1^N, y)$ is a truth table from $\mathcal{C}_n$, that is distributed according to $\mathcal{P}_n$. Furthermore, we say that $\mathcal{P}$ is polynomially samplable if $S(N) = \mathsf{poly}(N)$.

Remark 5.8. *For the special case where $\mathcal{C}$ is a class of fixed polynomial size circuits like $\mathsf{SIZE}[n^k]$ (or $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$) for any arbitrarily fixed k, we define a circuit representation scheme for $\mathcal{C}_n$ given by the set $R_n \subset \{0,1\}^{r(n)}$, where $r(n) = O(n^k \log n)$, such that every $\sigma \in R_n$ is mapped to a $\mathcal{C}$-circuit encoding of a function in $\mathcal{C}_n$. Note that this mapping is onto and each function in $\mathcal{C}_n$ could have many representations in R_n. We also assume that there exists a uniform circuit sequence in $\mathcal{C}$, which interprets this encoding as a $\mathcal{C}$-circuit and evaluates computations given this encoding.*

Now, we can define a distribution ensemble $\mathcal{P}$ over $\mathcal{C}$, where each $\mathcal{P}_n$ is a distribution over the $\mathcal{C}$-circuit encodings, which implicitly defines a distribution over $\mathcal{C}_n$ (under the mapping). We also define $S(r(n))$-samplability of $\mathcal{P}$, if there exists a randomised algorithm A running in time $S(r(n))$ such that for every $n \in \mathbb{N}$, $A(1^{r(n)}, y)$ is distributed identically to $\mathcal{P}_n$ over the random choice of $y \in \{0,1\}^{m(n)}$.

Definition 5.9 (**Average-case learnability** [BFKL93])**.** *Let $\mathcal{C}$ be a class of Boolean functions and $\mathcal{P} = \{\mathcal{P}_n\}$ be a distribution ensemble over $\mathcal{C}$. For any $0 < \varepsilon, \delta < 1/2$, we say that $\mathcal{C}$ is (ε, δ)-PAC-learnable in $T(n)$ time on average using random examples with respect to $\mathcal{P}$, if there exists a randomised algorithm $\mathcal{A}$ running in time at most $T(n)$ such that*

- *For every large enough n, for any fixed f drawn according to $\mathcal{P}_n$, for every $\mathcal{D}_n$ over $\{0,1\}^n$, $\mathcal{A}$ takes 1^n, a set of $m = m(n)$ labeled samples $(x_1, f(x_1)), \ldots, (x_m, f(x_m))$ where each $x_i \sim \mathcal{D}_n$, ε, δ and $w \in \{0,1\}^*$ (internal randomness) as its inputs, and outputs the description of a circuit h such that*

$$\Pr_{\substack{f \sim \mathcal{P}_n \\ w \in \{0,1\}^* \\ x_1, \ldots, x_m \sim \mathcal{D}_n}} \left\{ \Pr_{y \sim \mathcal{D}_n} \{h(y) = f(y)\} \geq 1 - \varepsilon \right\} \geq 1 - \delta$$

- *$\mathcal{A}$ runs in time at most $T(n)$.*

Furthermore, for any $0 < \varepsilon, \delta < 1/2$, we say that $\mathcal{C}$ is (ε, δ)-PAC-learnable in $T(n)$ time on average using random examples with respect to polynomially samplable distributions over $\mathcal{C}$, if there exists a learning algorithm $\mathcal{A}$ that runs in time $T(n)$ such that for every polynomially samplable distribution ensemble $\mathcal{P}$ over $\mathcal{C}$, we have that for every large enough n, $\mathcal{A}$ (ε, δ)-PAC-learns $\mathcal{C}_n$ on average using random examples with respect to $\mathcal{P}_n$.

We can naturally extend this definition to average-case learning $\mathcal{C}$ with respect to $\mathcal{P}$ using random examples drawn from a fixed distribution like U_n, or average-case PAC-learning $\mathcal{C}$ using membership queries.

5.2.2 Self-Reducibility

In our reductions, we use the following special properties of a function.

Definition 5.10 (Downward self-reducibility)**.** *A function $f_n : \{0,1\}^n \to \{0,1\}$ is downward-self-reducible if there is a deterministic polynomial time algorithm A such that for all $x \in \{0,1\}^n$, $A^{f_{n-1}}(x) = f_n(x)$.*

Definition 5.11 (Self-Correctibility)**.** *A function $f : \{0,1\}^n \to \{0,1\}$ is said to be self-correctible if there exists a constant $c \geq 0$ and a probabilistic polynomial-time algorithm A such that, for every large enough n, for any function $\mathcal{O} : \{0,1\}^n \to \{0,1\}$ that agrees with f_n on at least $(1-1/n^c)$-fraction of the inputs of length n, we have that $\Pr\{A^{\mathcal{O}}(x) = f_n(x)\} \geq 2/3$ for any $x \in \{0,1\}^n$.*

Babai et al. [BFNW93] show that any function g on n Boolean inputs can be transformed into a function g^* on n inputs from a large enough finite field, such that g^* coincides with g on the subset $\{0,1\}^n$. In particular, they show that

Theorem 5.12 ([BFNW93])**.** *There exists an* EXP*-Complete problem g^* which is self-correctible.*

Furthermore, Trevisan and Vadhan [TV07] construct a PSPACE-Complete problem which is based on a careful arithmetisation and padding of TQBF (using the interactive proof system for PSPACE), which has both these special properties.

Theorem 5.13 ([TV07])**.** *There exists a* PSPACE*-Complete language $f^* \in$ DSPACE$[n]$ that is both self-correctible and downward self-reducible (DSR).*

We also use the following collapse result.

Lemma 5.14. *If* EXP $\subseteq$ P/poly*, then* EXP $=$ PSPACE*. In particular, the function f^* (from Theorem 5.13) is complete for* EXP*.*

5.3 Unconditional Results for Hardness of Learning

Firstly, we show the hardness of learning BPE/poly over the uniform distribution using membership queries by randomised polynomial time algorithms. The proof of this result uses the following lemma from [KKO13].

Lemma 5.15. *Let $\mathcal{C}$ be any circuit class, $s : \mathbb{N} \to \mathbb{N}$ be a size function and f^* be the* PSPACE*-Complete problem from Theorem 5.13. There exists constant $c \in \mathbb{N}$ such that if $\mathcal{C}[s(n)]$ is learnable up to error n^{-c} in time $T(n)$, then at least one of the following holds:*

- $f^* \notin \mathcal{C}[s(n)]$.
- $f^* \in \mathsf{BPTIME}[\mathsf{poly}(T(n))]$.

Proof of Theorem 5.1. Towards a contradiction, assume that there exists constants $k, d \geq 1$ and a randomised learning algorithm $\mathcal{A}$ which learns $\mathsf{BPE/poly}$ in $O(n^d)$ time using membership queries over the uniform distribution, up to error $1/2 - 1/n^k$ and confidence $1/n$, for every large enough input length n. By non-uniformly fixing a good random string for the learner, we ensure that for some constant $c = c(d)$ and every function $g \in \mathsf{BPE/poly}$, A always outputs a hypothesis circuit of size $O(n^c)$ which computes g on at least $(1/2 + 1/n^k)$-fraction of n-length inputs. Thus, for every function in $\mathsf{BPE/poly}$, there exists a family of polynomial size circuits $\{h_n\}$ which $(1/2 + 1/n^k)$-approximates it, where h_i is the hypothesis output by the learner on input length i.

We next show that the existence of such a learner implies the existence of a function in BPE which cannot be $(1/2 + 1/n^k)$-approximated by polynomial sized circuits. Consider the PSPACE-Complete function f^* from Theorem 5.13 which is computable in time $\mathsf{DSPACE}[n]$. f^* is in $\mathsf{BPE/poly}$ (since f^* can be computed in E) and we use the learning algorithm for $\mathsf{BPE/poly}$ in Lemma 5.15 to see that $\mathsf{PSPACE} \subseteq \mathsf{BPP}$. Using a padding argument we observe that $\mathsf{DSPACE}[2^{O(n)}] \subseteq \mathsf{BPE}$. From Lemma 2.8, we see that there exists a function which cannot be $(1/2 + 1/n^k)$-approximated by circuits of size $n^{\log n}$. We can easily construct a Turing Machine which lexicographically searches for a truth table of a function on n inputs which cannot be $(1/2+1/n^k)$-approximated by $n^{\log n}$ sized circuits in $2^{O(n)}$ space and answers according to the first one it finds. From this we have that $\mathsf{DSPACE}[2^{O(n)}]$, and thus $\mathsf{BPE/poly}$ cannot be $(1/2 + 1/n^k)$-approximated by $n^{\log n}$ sized circuits, which leads to a contradiction. □

Remark 5.16. *[OS17] show that if for each c, a circuit class $\mathcal{C}[n^c]$ is $(1/2 - 1/n^c, 1/n)$-learnable using membership queries over the uniform distribution in $2^n/n^{\omega(1)}$ time, then for each k, there exists $L_k \in \mathsf{BPE}$ such that $L_k \notin \mathcal{C}[n^k]$ (Theorem 12). For any c, the idea of picking $\mathcal{C}[n^c] = \mathsf{SIZE}^{\mathsf{BPE}}[n^c]$ with linear-sized queries to BPE oracles and using the learning algorithm A which learns $\mathsf{BPE/poly}$ in their result to achieve a contradiction (as any function in BPE can be computed by constant sized $\mathsf{SIZE}^{\mathsf{BPE}}$-circuits with linear-sized oracle queries) does not work, as [OS17] crucially uses that $\mathcal{C}[n^c]$ has to be a subset of $\mathsf{SIZE}[n^{c'}]$ for some $c' = O(c)$.*

On the other hand, Theorem 4 in [FK09] shows that if $\mathcal{C}$ is learnable using membership queries over the uniform distribution in polynomial time then $\mathsf{BPE} \not\subseteq \mathcal{C}[\mathsf{poly}(n)]$. Proving Theorem 5.1 by setting $\mathcal{C}$ as $\mathsf{BPE/poly}$, again does not really work, as [FK09]'s result only holds true when $\mathcal{C} \subseteq \mathsf{P/poly}$, as it depends on the collapse of EXP to $\mathsf{P/poly}$.

Additionally, we also observe that the class E/poly cannot be learnt by deterministic learners using membership queries in $2^n/n$ time up to constant error over the uniform distribution. E/poly can equivalently be defined as the class of languages which can be computed by polynomial-sized circuit families with oracle access to some function in E, with the constraint that the oracle queries are of size $O(n)$.

We first rule out deterministic exact learners for E/poly in Angluin's model of learning [Ang88], i.e. the learners have access to a membership oracle, as well as an equivalence oracle, where the learner presents a hypothesis circuit to the equivalence oracle, and receives yes if the hypothesis exactly computes the target concept, or otherwise, receives a counter-example for the hypothesis.

Proposition 5.17. *There exists no deterministic exact learners for* E/poly *using membership queries and equivalence queries which run in time* $O(2^n/n)$.

Proof. The proof follows easily from [KKO13]. They show that if there exists a deterministic exact learner that learns a circuit class $\mathcal{C}[s(n)]$ in $2^n/n$ time using membership and equivalence queries, then there exists a function in E which cannot be computed by $\mathcal{C}$-circuits of size $s(n)$. Using E/poly for $\mathcal{C}$, we observe that there exists no deterministic exact learners which learn E/poly using membership queries and equivalence queries in $2^n/n$ time, because, otherwise, we would end up showing that there exists a function in E which cannot be computed by E/poly, which is trivially false. □

We extend this result to rule out any deterministic learners for E/poly, i.e. even learners which can output an approximate hypothesis.

Proposition 5.18. *For every constant* $\delta \in [0, 1/2 - 1/n)$, E/poly *is hard to learn up to error* δ *over the uniform distribution using membership queries by deterministic learning algorithms which run in time* $2^n/n$.

Proof. Towards a contradiction assume that there exists a constant $\delta > 0$ and a deterministic learner A running in time $T(n) = 2^n/n$ which learns E/poly using membership queries over the uniform distribution up to an error δ. For every n, let L be the language consisting of only those strings in $\{0,1\}^n$ which are *not* queried by A, when all of A's queries are answered with 0. Since A runs in time $T(n)$, it is the clear that $L \in \mathsf{E}$ and the size of $L^{-1}(0) \leq 2^n/n$. However, since A only queries those inputs which are not in L, it cannot distinguish between the all zeros function $\mathbf{0}$ and L. Indeed, if A returns a hypothesis h_1 which is $(1-\delta)$-close to $\mathbf{0}$, then h_1 agrees with L on at most a $\delta + 1/n$ fraction of the inputs. Similarly, if A returns a hypothesis h_2 which is $(1-\delta)$-close to L, then h_2 agrees with $\mathbf{0}$ on at most a $\delta + 1/n$ fraction. In either case, A can learn exactly one of L or $\mathbf{0}$, which contradicts our assumption that A learns every function in E/poly with error at most δ. □

Both propositions can also be extended to show similar results for unconditional hardness of learning $\mathsf{PSPACE/poly}$ by deterministic polynomial time learners.

5.4 Robustness of learning

We first establish the equivalences in Theorem 5.2 for hardness of learning $\mathsf{EXP/poly}$.

Lemma 5.19. *Let* $\mathsf{BPP} = \mathsf{EXP}$. *Then, for every* $c > 0$, $\mathsf{EXP/poly}$ *can be* $(1/n^c, 1/20n)$*-PAC-learnt using random examples in time polynomial in* n.

Proof. Firstly, we show that if $\mathsf{BPP} = \mathsf{EXP}$, then for every $c > 0$, we can $(1/n^c, 1/20n)$-learn $\mathsf{P/poly}$ in the worst-case using random examples over any arbitrary distribution $\mathcal{D}$. We consider the following search problem Π. On input $y = \langle 1^n, 1^{s(n)}, (x_1, b_1) \dots (x_m, b_m) \rangle$, where $x_i \in \{0,1\}^n$ and $b_i \in \{0,1\}$, if there exists a circuit C on n inputs of size at most $s(n)$ such that $C(x_i) = b_i$ for all $i \in [m]$, then $\Pi(y)$ outputs an encoding of C. The input length to Π is $t(n) = O(s(n) + n \cdot m(n))$. If $s(n) = \mathsf{poly}(n)$, then Π is in EXP as we can exhaustively search through all circuits on n inputs of size $s(n)$ and check if it is consistent with b_i on each $x_1, \dots, x_m$ in time $2^{O(s(n) \log s(n))} \cdot O(m(n) \cdot s(n)) = O\left(2^{t(n)} \mathsf{poly}(t(n))\right)$. Since $\mathsf{EXP} = \mathsf{BPP}$, there exists a randomised algorithm $\mathcal{A}$ which runs in time $\mathsf{poly}(t(n))$ over inputs of length $t(n)$, that outputs a circuit C_n of size at most $s(n)$ consistent with b_i on each $x_1, \dots, x_m$, if there exists one.

Following this, we use an argument based on Occam's razor [KV94b], to see that for every $\varepsilon, \delta > 0$ and $k \geq 0$, a consistent learner (a learner which always outputs a hypothesis which is correct on all the input examples) $\mathcal{A}$ (ε, δ)-learns $\mathsf{SIZE}[n^k]$ in the worst-case using random examples drawn from any fixed distribution $\mathcal{D}_n$ over $\{0,1\}^n$, if $m = O\left(\frac{1}{\varepsilon}(n^k \log n + \log\left(\frac{1}{\delta}\right))\right)$. For any fixed $c > 0$, when $\varepsilon = 1/n^c$ and $\delta = 1/20n$, $m(n) = O(n^{c+k+1})$. Thus, for every n, every $k \geq 0$ and every $f \in \mathsf{SIZE}[n^k]$, using $m = O(n^{c+k+1})$ many random examples drawn from $\mathcal{D}_n$ and their labels $f(x_1), \dots, f(x_m)$, the randomised algorithm $\mathcal{A}$ runs in time $\mathsf{poly}(n)$ and outputs a circuit of size $O(n^k)$ which is $(1-\varepsilon)$-close to f, with probability at least $(1-\delta)$.

Finally, observe that if $\mathsf{BPP} = \mathsf{EXP}$, then $\mathsf{EXP} \subseteq \mathsf{P/poly}$, which in turn implies that $\mathsf{EXP/poly} \subseteq \mathsf{P/poly}$. In other words, this means that there exists some constant $k \geq 0$ such that $\mathsf{EXP/poly} \subseteq \mathsf{SIZE}[n^k]$. This proves the lemma, as $\mathcal{A}$ can be used to $(1/n^c, 1/20n)$-learn $\mathsf{EXP/poly}$ in $\mathsf{poly}(n)$ time using random examples drawn from any fixed distribution $\mathcal{D}_n$. □

Lemma 5.20. *Let* $\mathsf{EXP} \neq \mathsf{BPP}$. *Then, there exists constant* $c > 0$ *such that* $\mathsf{EXP/poly}$ *is not* $(1/n^c, 1/20n)$*-learnable in the worst-case using random examples from the uniform distribution in time* $\mathsf{poly}(n)$.

Proof. Towards a contradiction assume that there exists a constant $a > 0$ and an $O(n^a)$-time learner $\mathcal{A}$ that $(1/n^c, 1/20n)$-learns $\mathsf{EXP/poly}$ using random examples over U_n, for all $c > 0$. We first show that the existence of the learner $\mathcal{A}$ for $\mathsf{EXP/poly}$ implies that $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Let g^* be an EXP-Complete problem which is self-correctible, whose existence is given by Theorem 5.12, with $c_1 \geq 0$ being the corresponding constant. Use $\mathcal{A}$ to $(1/n^{c_1}, 1/20n)$-learn g^* using random examples over the uniform distribution. Let $\mathcal{A}'$ be the algorithm which takes as input $y \in \{0,1\}^n$ in addition to the inputs of $\mathcal{A}$ and runs the learner $\mathcal{A}$, following which it returns the evaluation of the hypothesis circuit output by $\mathcal{A}$ on the input y. In other words, for every $n \in \mathbb{N}$, we have

$$\Pr_{\substack{w \in \{0,1\}^{r(n)} \\ x_1,\ldots,x_m \sim U_n}} \left\{ \Pr_{y \sim U_n} \{\mathcal{A}'(1^n, w, (x_1, g^*(x_1)), \ldots, (x_m, g^*(x_m)), y) = g^*(y)\} \geq 1 - 1/n^{c_1} \right\} \geq 1 - 1/20n$$

where both $r(n)$ and $m = m(n) = \mathsf{poly}(n)$.

By amplifying the correctness of $\mathcal{A}'$ using standard techniques, we can then non-uniformly fix the random strings $w, x_1, \ldots, x_m$ and the values of g^* on each x_i to get a polynomial sized circuit C, which takes input $y \in \{0,1\}^n$ and outputs the answer of $\mathcal{A}'$ on the advice string and y. Thus C_n agrees with g^* on at least $(1 - 1/n^{c_1})$-fraction of the inputs. Using C_n with the random self-reduction of g^* (and fixing another "good" random string non-uniformly in the resulting algorithm), we get a polynomial-sized circuit which computes g^* on every input and by the EXP-Completeness of g^*, we see that $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

Since $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we use Lemma 5.14 to observe that the PSPACE-Complete problem f^* given by Theorem 5.13 is now an EXP-Complete problem that is both downward self-reducible and self-correctible. Let c_2 be the constant associated with the self-corrector for f^*. For any integer k, given a procedure B_k which computes f^* on every instance of size k with high probability, we use $\mathcal{A}$ together with the downward self-reduction for f^*, followed by the self-corrector for f^* to obtain a procedure B_{k+1} that computes f^* on any input of size $k+1$. We use this inductively, to compute f^* on n inputs in probabilistic polynomial time.

More precisely, consider the following algorithm B_n which computes f^* on a given input $x \in \{0,1\}^n$ and does the following. First, it starts with a procedure B_{k_0}, for a constant k_0, which can be computed easily using a look-up table. Assuming that we have the procedure B_k for some input length $k \leq n$, we show how to construct the procedure B_{k+1} inductively. We use the learner $\mathcal{A}$ to learn the function f^*_{k+1} up to error $1/k^{c_2}$. For every input $f^*(y)$ passed to $\mathcal{A}$, where y is sampled from U_{k+1}, we use B_k in the downward self-reduction of f^* to compute $f^*(y)$. $\mathcal{A}$ outputs a hypothesis h_{k+1} which computes f^*_{k+1}

on at least a $(1 - 1/(k+1)^{c_2})$-fraction of the inputs with high probability. We now use the self-corrector for f^* to obtain from h_{k+1} a procedure B_{k+1} which is correct on every input of size $k+1$ with probability $1-\gamma$ (by using standard error reduction arguments), for some $\gamma > 0$ which we pick later. Repeating this process at most n times, we obtain B_n.

First, we show that B_n outputs $f^*(x)$ with probability at least $2/3$. Let $d(n)$ be the number of queries made by the DSR to the oracle f^*_{n-1} to compute $f^*(x)$. The idea is that at each stage k, B_k fails only if at least one of $m(n) \cdot d(n)$ queries answered by B_{k-1} is incorrect, which happens with probability at most $m(n)d(n)\gamma \leq 1/20n$ for $\gamma = 1/(20nm(n)d(n))$, or if $\mathcal{A}$ fails to output the right hypothesis, which happens with probability at most $1/20n$. Thus, the total failure probability at each stage is at most $1/10n$ and over the n stages, using the union bound, the total failure probability is at most $1/10 + \gamma \leq 1/3$.

Next, we observe that every stage B_k runs in time $\mathsf{poly}(k)$. It is easily seen that B_{k_0} runs in constant time. Assume that B_{k-1} runs in $\mathsf{poly}(k-1)$ time. At stage k, the time taken to compute f^* on $m(k)$ many inputs of length k is $O(m(k) \cdot d(k) \cdot \mathsf{poly}(k-1)) \leq \mathsf{poly}(k)$. After this, $\mathcal{A}$ takes $O(k^d)$ time to output h_k of size at most k^d, which is used by the $\mathsf{poly}(k)$-time self-corrector to compute f^* on all inputs of size k with high probability. Thus, B_k runs in time $\mathsf{poly}(k) \leq \mathsf{poly}(n)$. Since there at most n stages, the total running time of B_n is $\mathsf{poly}(n)$. This shows that $f^* \in \mathsf{BPP}$, contradicting the original assumption. □

Lemma 5.21. *Suppose that* $\mathsf{EXP/poly}$ *is* $(1/n^c, 1/20n)$*-learnable in the worst case for every constant* $c > 0$ *over the uniform distribution* U_n *using membership queries in time* $\mathsf{poly}(n)$*. Then,* $\mathsf{EXP} = \mathsf{BPP}$*.*

Proof Sketch. The proof is very similar to that of Lemma 5.20. Assume that there exists a $\mathsf{poly}(n)$-time learner $\mathcal{A}$ that $(1/n^c, 1/20n)$-learns $\mathsf{EXP/poly}$ over U_n for every $c > 0$, for every large enough input length n. We learn the function g^* (given by Theorem 5.12) using $\mathcal{A}$. Let h_n be a circuit of polynomial size output by $\mathcal{A}$ as the hypothesis, with probability at least $(1 - 1/20n)$. From the guarantees of the learner, we see that any h_n output by $\mathcal{A}$ computes g^* with error at most $1/n^{c_1}$ (after non-uniformly fixing its randomness). Let h'_n be the polynomial-sized circuit which uses h_n with the self-corrector for g^* (and non-uniformly another random string for the resulting algorithm) and computes g^* on length n. Thus, there exists a sequence of polynomial size circuits $\{h'_n\}_{n \in \mathbb{N}}$ which computes g^* on every large enough n. Since g^* is EXP-Complete, we see that $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

We now use $\mathcal{A}$ to learn the PSPACE-Complete problem f^* given by Theorem 5.13. Using similar ideas as Lemma 5.20, we see that $f^* \in \mathsf{BPP}$. Since EXP has now collapsed to PSPACE we see that $\mathsf{EXP} = \mathsf{BPP}$. □

Below, we show that learning EXP/poly on average using random examples, with respect to quasi-polynomially samplable distributions over EXP/poly, implies that $\mathsf{EXP} = \mathsf{BPP}$. See Remark 5.23 for a natural extension of this result to polynomially samplable distributions over EXP/poly.

Lemma 5.22. *Suppose that for every constant $c > 0$,* EXP/poly *is $(1/n^c, 1/20n)$-learnable on average using random examples from the uniform distribution U_n in time* poly(n)*, with respect to quasi-polynomially samplable distributions over* EXP/poly*. Then,* $\mathsf{EXP} = \mathsf{BPP}$*.*

Proof Sketch. Again, the proof strategy is very similar to that of Lemma 5.20. Let $\mathcal{A}$ be a poly(n)-time learner such that for every quasi-polynomially samplable distribution ensemble $\mathcal{P}$ over EXP/poly, $\mathcal{A}$ $(1/n^c, 1/20n)$-learns EXP/poly using random examples from U_n, with respect to $\mathcal{P}$, for all $c > 0$. Formally, we have

$$\Pr_{\substack{f\sim\mathcal{P}_n \\ w\in\{0,1\}^* \\ x_1,\ldots,x_m\sim U_n}} \{\mathcal{A}(1^n, \varepsilon, \delta, w, x_1, f(x_1), \ldots, x_m, f(x_m)) \text{ is } (1-\varepsilon)\text{-close to } f\} \geq 1 - \delta$$

for sufficiently large n.

We first show that the existence of $\mathcal{A}$ implies $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Indeed, consider the quasi-polynomially samplable distribution $\mathcal{P}$ supported only on the function g^* given by Theorem 5.12, defined by the algorithm $A_{\mathcal{P}}$ which takes as inputs 1^N and outputs the truth table of g^* on n inputs by running the EXP-machine which computes g^* on every input in $\{0,1\}^n$. Clearly, the running time of the sampler is quasi-polynomial in N. From our assumption in the lemma and since $\mathcal{P}$ is supported only on g^*, $\mathcal{A}$ outputs a hypothesis which is $(1 - 1/n^{c_1})$-close to g^* with probability at least $1 - 1/20n$. Using the same idea as Lemma 5.20, we show that $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

Following this, we design a similar quasi-polynomially samplable distribution $\mathcal{D}$ supported on the PSPACE-Complete problem f^* given by Theorem 5.13. Since $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we see that f^* is also EXP-Complete. Again, since $\mathcal{D}$ is supported only on f^*, $\mathcal{A}$ can be used to output a hypothesis which approximates f^* with high probability and following from the ideas used earlier, we show that $f^* \in \mathsf{BPP}$ proving that $\mathsf{EXP} = \mathsf{BPP}$. □

Using the proof strategy of Lemma 5.21, we can also show a natural extension of Lemma 5.22 for average-case learning using membership queries as well.

Remark 5.23. *We also extend lemma 5.22 to the case where the samplable distribution is over* $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$*-circuit encodings in* $R_n \subseteq \{0,1\}^{r(n)}$*, where* $r(n) = O(n^{2k+1})$*. In such a case, we only need* $\mathsf{poly}(n)$*-time learnability of* $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$ *using random examples over* U_n *with respect to* ***polynomially samplable*** *distributions over* $\mathcal{C}_n$*, to show that* $\mathsf{EXP} = \mathsf{BPP}$*.*

Indeed, for the distribution $\mathcal{P}$ *over* $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$ *supported only on the function* g^* *(or* f^**) used in the proof, the sampler* $A'_{\mathcal{P}}$ *takes* 1^{n^k} *as input and outputs a* $\mathsf{SIZE}^{\mathsf{EXP}}[n^k]$*-circuit encoding of* g^* *which is just an* EXP*-oracle gate on* n *inputs and this encoding is of size* $O(n^2)$*. The running time of this sampling algorithm is polynomial in the input size.*

Using these lemmas, we now prove Theorem 5.2.

Proof of Theorem 5.2. We only establish equivalences with respect to strong learning, since the AdaBoost algorithm [FS97] shows an equivalence between weak and strong learning. In other words, from any polynomial time $(1/2 - 1/n^c, 1/10n)$-PAC-learner using random examples (or membership queries), we can get a polynomial time $(1/n^c, 1/10n)$-PAC-learner using random examples (or membership queries), by application of the boosting algorithm. The following implications establish the desired equivalences.

$(b) \implies (a), (c) \implies (a)$: The contrapositives of each of these implications follow from Lemma 5.19. In particular, PAC-learning $\mathsf{EXP/poly}$ with error at most $1/n^c$ using random examples, implies PAC-learnability of $\mathsf{EXP/poly}$ using membership queries, where the queries are just made on the random examples given to the learner.

$(d) \implies (b), (e) \implies (c)$: Follows from the definitions, since PAC-learning $\mathsf{EXP/poly}$ in the worst case in $\mathsf{poly}(n)$ time using random examples implies PAC-learnability for $\mathsf{EXP/poly}$ on average in $\mathsf{poly}(n)$ time using random examples, for any distribution over $\mathsf{EXP/poly}$. A similar implication holds for learning with membership queries too.

$(a) \implies (b)$: Suppose $\mathsf{EXP/poly}$ is $(1/n^c, 1/10n)$ PAC-learnable in polynomial time using random examples over every arbitrary distribution, for any c. In particular, this means that $\mathsf{EXP/poly}$ can be $(1/n^c, 1/10n)$-learnt in polynomial time using random examples over the uniform distribution. The implication follows from the contrapositive of Lemma 5.20.

$(a) \implies (c)$: Similar to the previous implication, we see that $\mathsf{EXP/poly}$ is $(1/n^c, 1/10n)$-learnable in polynomial time using membership queries over the uniform distribution. The implication holds from the contrapositive of Lemma 5.21.

$(a) \implies (d), (a) \implies (e)$: The implications follow from Lemma 5.22 (and Remark 5.23) and its corresponding extension to learning on average with membership queries. □

5.5 Reducing Succinct Search to Decision

The key concepts in this section are verifiability and succinct search. We define verifiers first.

Definition 5.24. *Given language $L \subseteq \{0,1\}^*$ and polynomial-time computable relation $V(\cdot,\cdot)$, we say that V is a verifier for L if for each $x \in \{0,1\}^*$, $x \in L$ iff $\exists y V(x,y)$.*

Given language L, a verifier V for L, and function $f : \mathbb{N} \to \mathbb{N}$, we say that L has $f(n)$-size proofs with respect to V, such that for each $x \in \{0,1\}^*$, $x \in L$ implies $\exists y, |y| \leq f(|x|) : V(x,y)$. We say that L has $f(n)$-size proofs if there is a verifier V for L such that L has $f(n)$-size proofs with respect to V.

Given language L, a verifier V for L and a machine class $\mathcal{D}$, we say that L has $\mathcal{D}$-computable proofs with respect to V if there is a machine $M \in \mathcal{D}$ such that for each $x \in \{0,1\}^*$, $x \in L$ implies $V(x, M(x))$. We say that L has $\mathcal{D}$-computable proofs if there is a verifier V for L such that L has $\mathcal{D}$-computable proofs with respect to V.

Note that NP is the class of languages with polynomial-sized proofs, NEXP is the class of languages with exponential-sized proofs, and for $\mathcal{D} \in \{\mathsf{EXP}, \mathsf{PSPACE}\}$, $\mathcal{D}$ is the class of languages with $\mathcal{D}$-computable proofs (where we abuse notation and use $\mathcal{D}$ to refer both to a machine class and to the set of languages computable by such machines).

Next we define succinct search. We will assume w.l.o.g that the proof size for any verifier is a power of 2 – this can be ensured by padding the proof if necessary.

Definition 5.25. *Given language L and verifier V for L, we say that **succinct search is easy** for L with respect to V if there exists a probabilistic polynomial-time machine N such that for each $x \in L$, there is a V-proof y such that with probability $1 - o(1)$, $N(x)$ outputs a circuit C for which $\mathsf{tt}(C) = y$.*

Thus succinct search is easy for L with respect to a verifier V if there is a probabilistic polynomial-time machine outputting compressed descriptions of V-proofs with high probability for any positive instance of L.

Using the downward self-reducibility of SAT, it is straightforward to see that $\mathsf{NP} \subseteq \mathsf{BPP}$ iff for each $L \in \mathsf{NP}$ and for every verifier V such that L has poly-size proofs with respect to V, succinct search is easy for L with respect to V. We now show analogous results for $\mathsf{PSPACE}, \mathsf{EXP}$ and NEXP. First we show for each of these classes that easiness of the class implies easiness of succinct search.

We need the Easy Witness Lemma of Impagliazzo, Kabanets and Wigderson [IKW02].

Lemma 5.26 (Easy Witness Lemma [IKW02])**.** *If $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, then for each $L \in \mathsf{NEXP}$ and for each verifier V for L such that L has exponential-size proofs with respect to V, for each $x \in L$, there is a polynomial-size circuit C_x such that $V(x, \mathsf{tt}(C_x))$ holds.*

Lemma 5.27. *The following implications hold:*

1. *Let $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$. If $\mathcal{D} = \mathsf{BPP}$, then for each $L \in \mathcal{D}$ and for each verifier V such that L has $\mathcal{D}$-computable proofs with respect to V, succinct search is easy for L with respect to V.*

2. *If $\mathsf{NEXP} = \mathsf{BPP}$, then for each $L \in \mathsf{NEXP}$ and for each verifier V such that L has exponential-size proofs with respect to V, succinct search is easy for L with respect to V.*

Proof. We establish the first item. Let $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$, and assume $\mathcal{D} = \mathsf{BPP}$. Let $L \in \mathcal{D}$ and V be a verifier for L such that L has $\mathcal{D}$-computable proofs with respect to V. We construct a probabilistic poly-time machine N such that for each input $x \in L$, there is a V-proof w such that with high probability $\mathsf{tt}(N(x)) = w$. Let M be a $\mathcal{D}$-machine outputting V-proofs for positive instances of L.

Consider the language $L' = \{\langle x, i\rangle \mid i^{\text{th}} \text{ bit of } M(x) \text{ is } 1\}$. Since M is a $\mathcal{D}$ machine, we have that $L' \in \mathcal{D}$. By assumption, $\mathcal{D} = \mathsf{BPP}$, therefore there is a probabilistic poly-time machine N' deciding L'. Assume w.l.o.g that N' has error at most $2^{-|y|^2}$ on any input y. Given input x, N operates as follows. It first computes a probabilistic poly-size circuit C' simulating N'. This can be done using the standard efficient conversion of efficient algorithms into small circuits. It then hardwires x into the first part of the input for C', obtaining a circuit C'_x. It then fixes the random input of the circuit C'_x to a uniformly generated random string r to obtain a circuit $D'_{x,r}$, which it outputs. Since the error of N' is smaller than $2^{-|y|^2}$ on any input y, by a simple union bound, with probability $1 - o(1)$ over the choice of the random string r, $D'_{x,r}$ correctly computes the i^{th} bit of $M(x)$ for each $i \in [m]$. For $x \in L$, $V(x, M(x))$ holds, and therefore N efficiently solves succinct search for L with respect to V.

We establish the second item. Assume $\mathsf{NEXP} = \mathsf{BPP}$ and let $L \in \mathsf{NEXP}$ and V be a verifier for L such that L has exponential-size proofs with respect to V. Since $\mathsf{NEXP} = \mathsf{BPP}$, we have that $\mathsf{NEXP} \subseteq \mathsf{P/poly}$. By Lemma 5.26, there is a polynomial p such that for each $x \in L$, there is a circuit C_x of size at most $p(|x|)$ such that $V(x, \mathsf{tt}(C_x))$ holds.

Consider the language $L' = \{\langle x, i\rangle \mid \text{There exists circuit } C_x \text{ of size } p(|x|) \text{ such that } V(x, \mathsf{tt}(C_x)) \text{ is 1, and the } i^{\text{th}} \text{ bit of the lexicographically first such circuit is 1}\}$. Clearly $L' \in \mathsf{EXP}$, just by enumerating circuits of size $p(|x|)$ in lexicographic order and finding the first one encoding a V-proof for x, if one exists. Since $\mathsf{EXP} = \mathsf{BPP}$, there is a probabilistic poly-time machine N' deciding L' with error exponentially small. We construct a probabilistic poly-time machine N as follows: on input x, N runs N' on $\langle x, i\rangle$ for each i at most the description length of a circuit of size $p(|x|)$. It outputs the circuit C whose

description has its i^{th} bit set to 1 iff N' accepts $\langle x, i\rangle$. Since N' has error exponentially small, we have that with error exponentially small, N outputs a circuit C encoding a V-proof of x, and therefore N efficiently solves succinct search for L with respect to V. □

For the reverse directions, we use the PCP characterisation of NEXP [BFL91, FRS94], where we only require a polynomial upper bound on query complexity of the verifier.

Theorem 5.28 (PCPs for NEXP [BFL91, FRS94])**.** *Let $L \in \mathsf{NEXP}$. There is a probabilistic poly-time oracle machine V' such that:*

1. *For each $x \in L$, there is y of length exponential in $|x|$ such that $V'(x)$ accepts with probability at least $2/3$ when given oracle access to y.*
2. *For each $x \notin L$ and for all y, $V'(x)$ accepts with probability at most $1/3$ when given oracle access to y.*

We now show that easiness of succinct search implies easiness of decision for any $L \in \mathsf{NEXP}$.

Lemma 5.29. *Let $L \in \mathsf{NEXP}$ and V be any verifier such that L has exponential-size proofs with respect to V. If succinct search is easy for L with respect to V, then $L \in \mathsf{BPP}$.*

Proof. Let $L \in \mathsf{NEXP}$. We show that $L \in \mathsf{BPP}$. By Theorem 5.28, there is a probabilistic poly-time oracle machine V' such that if $x \in L$, there is y of length exponential in $|x|$ for which V' accepts with high probability on x when given oracle access to y, and if $x \notin L$ rejects with high probability irrespective of the oracle. Let $r(|x|) = \mathsf{poly}(|x|)$ be the number of random bits used by V'.

Now consider a verifier V for L which given input x and proof $y' = \langle y, 1^{2^{r(|x|)}}\rangle$ (in other words, y is padded with the total number of random choices; proof size is still exponential in $|x|$), accepts iff y' is in the specified format and $V'(x)$ accepts with oracle y on a majority of its computation paths. Since succinct search is easy for L with respect to V, there is a probabilistic poly-time machine N such that for input $x \in L$, there is a V-proof y' for x such that with high probability $\mathsf{tt}(N(x)) = y'$. We define a probabilistic poly-time machine W that on input x simulates $V'(x)$ as follows. It first runs $N(x)$ to find a circuit C. It then runs $V'(x)$, answering all oracle calls to y by simulating C on input corresponding to the bit of y' that is queried (after suitably padding the oracle query to y). It accepts iff $V'(x)$ accepts.

If $x \in L$, by using the assumption that N solves succinct search, $W(x)$ accepts with probability close to $2/3$. If $x \notin L$, $W(x)$ rejects with probability close to $2/3$ since the circuit C output by $N(x)$ corresponds to some purported V'-proof, and every such V'-proof is rejected with high probability by V when given oracle access to the proof. □

Theorem 5.30. *The following statements are true.*

1. *Let* $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$. $\mathcal{D} = \mathsf{BPP}$ *iff for each* $L \in \mathcal{D}$ *and for each verifier* V *for* L *such that* L *has* $\mathcal{D}$*-computable proofs with respect to* V*, succinct search is easy for* L *with respect to* V*.*

2. $\mathsf{NEXP} = \mathsf{BPP}$ *iff for each* $L \in \mathsf{NEXP}$ *and for each verifier* V *such that* L *has exponential-size proofs with respect to* V*, succinct search is easy for* L *with respect to* V*.*

Proof. The forward directions of both items follow from Lemma 5.27. The backward direction of the second item follows Lemma 5.29. The backward direction of the first item follows from Lemma 5.29 and the fact that for $\mathcal{D} \in \{\mathsf{PSPACE}, \mathsf{EXP}\}$, if $L \in \mathcal{D}$ and V is a verifier for L such that L has $\mathcal{D}$-computable proofs with respect to V, then $L \in \mathsf{NEXP}$ and L has exponential-size proofs with respect to V. □

Theorem 5.31. *The following results hold.*

1. $\mathsf{R_{Kt}} \in \mathsf{BPP}$ *iff for each verifier* V *for* $\mathsf{R_{Kt}}$ *such that* $\mathsf{R_{Kt}}$ *has* EXP*-computable proofs with respect to* V*, succinct search is easy for* $\mathsf{R_{Kt}}$ *with respect to* V*.*

2. $\mathsf{R_{KS}} \in \mathsf{BPP}$ *iff for each verifier* V *for* $\mathsf{R_{KS}}$ *such that* $\mathsf{R_{KS}}$ *has* PSPACE*-computable proofs with respect to* V*, succinct search is easy for* $\mathsf{R_{KS}}$ *with respect to* V*.*

Proof. The backward directions of both items follow from Lemma 5.29 and the facts that $\mathsf{R_{Kt}}$ and $\mathsf{R_{KS}}$ are in NEXP.

For the forward direction of the first item, we use the result shown in [ABK+06] (Section 5) that $\mathsf{R_{Kt}} \in \mathsf{BPP}$ implies $\mathsf{EXP} = \mathsf{BPP}$. Combining this with the first item of Lemma 5.27 completes the proof.

For the forward direction of the second item, we use the result shown in [ABK+06] (Theorem 33) that $\mathsf{R_{KS}} \in \mathsf{BPP}$ implies $\mathsf{PSPACE} = \mathsf{BPP}$. Combining this with the first item of Lemma 5.27 completes the proof. □

5.6 Barriers for Conditional Hardness of Learning

Firstly, we formally define what it means to have a Black-Box Turing reduction from a language L to a PAC-learning algorithm for a class $\mathcal{C}$. Fix the error of the learner to be $\varepsilon = 1/\mathsf{poly}(n)$ (we ignore the confidence parameter, but this only makes our hardness results stronger).

Definition 5.32 (Turing Reduction to Learning $\mathcal{C}$.)**.** *A B-adaptive black-box reduction from deciding L to PAC-learning $\mathcal{C}$ using random examples up to error ε, is a tuple of probabilistic polynomial time algorithms $R = (T_1, \ldots, T_B, M)$ where R is given an input $z \in \{0,1\}^n$ and randomness $w \in \{0,1\}^*$. For each query, R constructs a joint distribution $(X, f(X))$ over $\{0,1\}^r \times \{0,1\}$ for some $r \leq n$ and $f \in \mathcal{C}$, samples a set $S = \{(x_i, f(x)_i)\}_{i \leq \mathsf{poly}(n)}$ of independent labeled examples according to $(X, f(X))$ and passes it to the learner. Let $t(n)$ be the query complexity of each round of adaptivity. R decides z by doing the following*

- *For each $1 \leq j \leq B$, T_j gets input z, fresh random bits from w and all the $(j-1)\cdot t(n)$ hypothesis circuits answered for the queries from the previous rounds (T_1 only has z and randomness w as input), and outputs $t(n)$ new queries $S_{j1}, \ldots, S_{jt}$ to the learner, each of which are sets of labeled examples independently sampled from joint distributions $(X_{j1}, Y_{j1}), \ldots, (X_{jt}, Y_{jt})$.*
- *R only has oracle access to the learner.*
- *M takes as input z, fresh random bits from w and the $B \cdot t(n)$ hypothesis circuits which are the answers given by the learner for all the queries asked by $T_1, \ldots, T_B$, and outputs the answer.*
- *The reduction guarantees that if for every oracle A that is a $\mathcal{C}$-circuit learner, if every hypothesis circuit returned by the learner is $(1-\varepsilon)$-close with respect to its corresponding query made in $T_1, \ldots, T_B$, then $M(z) = L(z)$ with high probability over the internal randomness of the reduction R.*

Definition 5.33. *For any B-adaptive black-box reduction $R = (T_1, \ldots, T_B)$ from deciding L to PAC-learning $\mathcal{C}$ using random examples up to error ε, we have*

- *R is called **strongly black-box**, if $T_1, \ldots, T_B, M$ only have oracle access to the hypothesis circuits, and M decides L for any $(1-\varepsilon)$-close hypothesis circuit corresponding to each query made by $T_1, \ldots, T_B$.*
- *If $B = 1$, we call the reduction as **non-adaptive** and if R is strongly black-box and M makes only non-adaptive queries to the hypothesis circuits, we call the reduction as **fully non-adaptive**.*
- *R is **oblivious**, if $T_1, \ldots, T_B$ output new queries using only fresh randomness from w as input and access to the hypotheses generated during the previous rounds. Furthermore, M accesses each hypothesis using non-adaptively generated, identically*

distributed samples made from the corresponding distribution over which each hypothesis is guaranteed to be a good approximation. In particular, the obliviousness of the reduction means that the queries to the learner do not depend on the input z.

Unless mentioned we think of the query complexity $t(n) = \mathsf{poly}(n)$. It is worth to note that since the algorithms $T_1, \ldots, T_B$ are polynomial time algorithms, each joint distribution (X, Y) must be efficiently samplable.

We first prove Lemma 5.5. This is a reformulation of the proof of Lemma 5.20 as a black-box reduction from a PSPACE-Complete language to learning PSPACE/poly over the uniform distribution.

Proof of Lemma 5.5. This a re-adaptation of the proof of Corollary 5.3. Consider $R = (T_1, \ldots, T_n, M)$ as an n-adaptive reduction from deciding f^* to learning PSPACE/poly using random examples over the uniform distribution, where $T_1, \ldots, T_n, M$ are probabilistic polynomial time algorithms which are defined as follows.

For every $k \leq n$, T_k makes exactly one query to the learner which is the set of examples $S_k = \{(x_i, y_i)\}_{i \leq \mathsf{poly}(n)}$ drawn from the joint distribution $(U_k, f^*(U_k))$, where U_k is the uniform distribution over $\{0,1\}^k$. In the k^{th} round of adaptivity, T_k only makes oracle queries to the hypothesis h_{k-1} output in the last round. Indeed, let h'_{k-1} be the oracle circuit which uses h_{k-1} as an oracle in the self-corrector algorithm for f^*, and computes f^* on all $k-1$ length inputs with high probability. It then outputs a set S_k of independent labeled samples (x_i, y_i), where each x_i is sampled uniformly at random from U_k and $y_i = f^*(x_i)$ computed by using the downward self-reducibility of f^* with h'_{k-1}. M takes the final hypothesis h_n output by the learner over n inputs and outputs the value of the self-corrector of f^* with the oracle h_n. The correctness of R and the run-time analyses of $T_1, \ldots, T_n, M$ follow from the proof techniques of Lemma 5.20.

We next show that R has the required properties. As the self-corrector and the downward self-reduction for f^* work for any oracle which satisfy the appropriate constraints, R is correct for any oracle which outputs *any* correct hypothesis for f^* with respect to the uniform distribution (over different input lengths). Further, it makes only oracle queries to the learner, as well as to all the hypothesis circuits $h_1, \ldots, h_n$. This makes the reduction strongly black-box. By the property of the self-corrector, M only makes queries sampled from U_n to h_n, which is the same as the query made to the learner. The obliviousness now follows, since only f^* is learnt in each query, irrespective of the choice of z. □

Our main result is the following.

Theorem 5.34. *There exists a universal constant $c > 0$ such that the following holds. For any language L, $\varepsilon_0 = 1/n^c$ and any $B = \mathsf{poly}(n)$, if there exists an oblivious, B-adaptive, strongly black-box reduction from L to learning $\mathsf{NP/poly}$ using random examples over polynomially samplable distributions up to error ε_0, then $\overline{L} \in \mathsf{AM}^{\mathsf{poly}}$.*

Recall that the class $\mathsf{AM}^{\mathsf{poly}}$ refers to the class of languages recognised by constant-round interactive protocols with advice, where we require proper acceptance/rejection probabilities only when the advice is correct. [FF93] show that $\mathsf{AM}^{\mathsf{poly}} = \mathsf{NP/poly}$. Using Theorem 5.34 with $L = \mathsf{SAT}$, we get

Corollary 5.35. *There exists a universal constant $c > 0$ such that the following holds. For $\varepsilon_0 = 1/n^c$ and any $B = \mathsf{poly}(n)$, if there exists an oblivious, B-adaptive, strongly black-box reduction from deciding SAT to learning $\mathsf{NP/poly}$ using random examples from polynomially samplable distributions up to error ε_0, then $\mathsf{coNP} \subseteq \mathsf{NP/poly}$.*

Corollary 5.35 implies Theorem 5.6 easily, since $\mathsf{coNP} \subseteq \mathsf{NP/poly}$ implies $\Sigma_3^P = \Pi_3^P$ [Yap83].

We now prove Theorem 5.34. For ease of presentation, we prove it for the case of an oblivious, B-adaptive, strongly black-box reduction $R = (T_1, \ldots, T_B, M)$ from L to learning $\mathsf{NP/poly}$ over the *uniform distribution*. This means that the queries generated by the reduction to the learner are labeled examples drawn from the uniform distribution. The proof of Theorem 5.34 can be easily extended to the case where the reduction makes queries to learn over any polynomially samplable distribution, rather than just the uniform distribution.

Proof of Theorem 5.34. Let $R = (T_1, \ldots, T_B, M)$ be an oblivious, B-adaptive, strongly black-box reduction from L to learning $\mathsf{NP/poly}$ using random examples over the uniform distribution, where $T_1, \ldots, T_B, M$ are probabilistic polynomial time machines.

By using standard techniques (Adleman's trick over R), we non-uniformly fix a random string w_1 to be used by $T_1, \ldots, T_B$, as advice to R. This ensures that for every input $z \in \{0,1\}^n$, R decides z correctly with probability at least $1 - \gamma$ over M's randomness, for some $0 < \gamma < 1/2$.

By fixing w_1 non-uniformly, we also ensure that the queries made by the first round of adaptivity T_1, get fixed. Let $(X_{11}, Y_{11}), \ldots, (X_{1t}, Y_{1t})$ be the joint distributions which are constructed by T_1, where for each $i \in [t]$, $X_{1i} = U_{r_{1i}}$, the uniform distribution over r_{1i} bits for $r_{1i} \leq n$ and $Y_{1i} = f_{1i}(U_{r_{1i}})$ for some $f_{1i} \in \mathsf{NP/poly}$. Define $Q_1 = \{(r_{11}, f_{11}), \ldots, (r_{1t}, f_{1t})\}$.

For any $b \in [B]$, assume that the queries to the learner from the previous rounds have been fixed as Q_{b-1}. T_b takes fresh randomness from w_1 (which has non-uniformly

been fixed) as input and has oracle access to the hypotheses, which $(1-\varepsilon_0)$-approximate each queried function in Q_{b-1} according to the uniform distribution over their respective input lengths. Let $\mathcal{S}_b$ be the set of all tuples of joint distributions which can potentially be queried by T_b to the learner based on the choices it makes after w_1 and the hypothesis oracles for the queries made up to T_{b-2} have been fixed. Arbitrarily pick any such tuple $((U_{r_{b1}}, f_{b1}(U_{r_{b1}}), \ldots, (U_{r_{bt}}, f_{bt}(U_{r_{bt}}))$ and fix $Q_b = Q_{b-1} \cup \{(r_{b1}, f_{b1}), \ldots, (r_{bt}, f_{bt})\}$. In other words, this implicitly means that there exist some choice of hypothesis oracles, each of which $(1-\varepsilon_0)$-approximates the functions queried by T_{b-1}, such that along with the hypothesis oracles fixed in the previous rounds, T_b generates the tuple $((U_{r_{b1}}, f_{b1}(U_{r_{b1}}), \ldots, (U_{r_{bt}}, f_{bt}(U_{r_{bt}}))$.

Let $\ell = Bt$ be the total number of learner queries made. For $i \in [\ell]$, let $\{D^i_n\}$ be the sequence of non-deterministic circuits which verifies f_i. Let p_i be the probability that f_i accepts a string sampled from U_{r_i}, i.e. $\left|p_i - \Pr_{y \sim U_{r_i}}\{h_i(y) = 1\}\right| \leq \varepsilon_0$. Finally, let q be the number of *non-adaptive* queries M makes to each hypothesis h_i according to the distribution U_{r_i}.

On input $z \in \{0,1\}^n$, non-uniform advice $D^1_{r_1}, \ldots, D^\ell_{r_\ell}, p_1, \ldots, p_\ell$, and any error parameter $0 < \delta < 1/2$, the Arthur-Merlin protocol for $\overline{L}$ is as follows :

1. **Verifier:** Let $K = \frac{4q^2\ell^2}{\delta^2} \ln\left(\frac{2q\ell}{\delta}\right)$. Run M independently K times. For each $k \in [K]$, let $V^k = (v^k_{11}, \ldots, v^k_{1q}), (v^k_{21}, \ldots, v^k_{2q}), \ldots, (v^k_{\ell 1}, \ldots, v^k_{\ell q})$ be the set of queries made by M in the k^{th} run, where for every $i \in [t]$, $(v^k_{i1}, \ldots, v^k_{iq})$ are the q queries of length r_i made to h_i. Send all the queries $V^1, \ldots, V^K$ to the prover.

2. **Prover:** For each v^k_{ij}, respond by saying if $v^k_{ij} \in f_i$ and if so, provide a certificate that $v^k_{ij} \in f_i$ which can be verified by $D^i_{r_i}$.

3. **Verifier:** Accept if all the conditions hold :

   - All certificates sent by the prover are valid (verified by the circuits $D^1_{r_1}, \ldots, D^t_{r_\ell}$).
   - For every $1 \leq i \leq \ell$, at least $q \cdot \left(p_i K - \varepsilon_0 K - \left(\sqrt{K \ln\left(\frac{2\ell q}{\delta}\right)}\right)\right)$ queries made to the prover for f_i are answered by the prover as "yes".
   - For k picked uniformly at random from $[K]$, on input z and using the answers given by the prover for the oracle queries in V^k, the output of the k^{th} run of M is NO.

Completeness: Let $\varepsilon = \gamma + 2\delta + 2q\ell\varepsilon_0$. Pick $c > 0$ large enough, such that $\varepsilon_0 = 1/n^c$ makes $\varepsilon = 1/\mathsf{poly}(n)$. To show completeness, for any input $z \in \overline{L}$, observe that an honest prover would send correct answers for every query and corresponding certificates,

if necessary. Furthermore, if all queries are answered correctly, then M decides z correctly with probability at least $1-\gamma$ on every run. Finally, observe that for each fixed $i \in [\ell], j \in [q]$, the queries $v_{ij}^1, \ldots, v_{ij}^K$ are independent and distributed identically according to U_{r_i}, with probability at least $p_i - \varepsilon_0$ of being a yes instance. Using Hoeffding's inequality (Theorem 2.7), with probability at least (1-$\frac{\delta}{q\ell}$), at least $p_i K - \varepsilon_0 K - \left(\sqrt{K \ln(2q\ell/\delta)}\right)$ of these queries are yes instances. By a union bound, with probability at least $(1-\delta/\ell)$, this is satisfied for all $j \in [q]$ and at least $q \cdot \left(p_i K - \varepsilon_0 K - \left(\sqrt{K \ln(2q\ell/\delta)}\right)\right)$ many queries made to f_i are yes instances. Using a final union bound over all $1 \leq i \leq \ell$, we see that with probability at least $(1-\delta)$, the threshold is satisfied for every function f_i. Thus, the verifier accepts z with probability at least $1 - \delta - \gamma \geq 1 - \varepsilon$.

Soundness: For any $z \notin \overline{L}$, note that the cheating prover can only cheat by saying "no" on a query which is a yes instance for any f_i, which is ensured by the first condition. Using Hoeffding's inequality in the same way as above, we see that with probability at least $(1-\delta)$, for every $1 \leq i \leq \ell$, at most $q\left(p_i K + \varepsilon_0 K + \left(\sqrt{K \ln(2q\ell\delta)}\right)\right)$ many queries made to the prover for f_i are yes instances. In particular, this means that with probability at least $(1-\delta)$, the verifier ensures that the prover can cheat on at most $2q \cdot \left(\sqrt{K \ln(2q\ell/\delta)} + \varepsilon_0 K\right)$ many yes instances for each f_i. Thus, the probability that there exists a run of M which consists of a query to some f_i on which the prover has cheated is at most $2q\ell \cdot \left(\sqrt{\ln(2q\ell/\delta)/K} + \varepsilon_0\right) \leq \delta + 2q\ell\varepsilon_0$ for the value of K defined. Thus, the overall probability that the verifier accepts is at most $\delta + \gamma + (\delta + 2q\ell\varepsilon_0) \leq \gamma + 2\delta + 2q\ell\varepsilon_0 \leq \varepsilon$. □

Remark 5.36. *In addition, we can also extend the proof to the case where M still makes non-adaptive queries but is not constrained distributionally in its access to all the hypotheses, by directly applying the techniques of [BT06] for the simulation of R in* $\mathsf{AM}^{\mathsf{poly}}$.

5.7 Open Questions

Can we show the hardness of PAC-learning $\mathsf{NP/poly}$ efficiently using random examples assuming that $\mathsf{NP} \neq \mathsf{BPP}$? A potential direction is to consider non black-box reductions for the NP-hardness of PAC-learning $\mathsf{NP/poly}$. This viewpoint has lent itself some success in the case of worst-case to average-case reductions [GST07, Hir18, HW19, Hir20a, Hir20b] and in our case, hardness of efficiently learning $\mathsf{EXP/poly}$. Indeed, the reduction for $\mathsf{EXP/poly}$ only works if the learning algorithm runs in polynomial time, although the reduction still uses the learning algorithm as an oracle.[8] Moreover, the proof of Lemma

[8] We need EXP to collapse to PSPACE for the reduction to succeed. An efficient learner for $\mathsf{EXP/poly}$ outputs polynomial-sized hypothesis circuits for any language in EXP. Using standard techniques, this implies $\mathsf{EXP} \subseteq \mathsf{P/poly}$, which leads to the required collapse.

3.6 from [CHO+20] shows a non black-box reduction from an approximate version of MCSP to learning P/poly by sub-exponential-sized circuits (and thus, learning NP/poly too).[9] Note that, it is unclear if approximate MCSP is NP-Hard and this reduction does not imply the NP-Hardness of PAC-learning NP/poly efficiently.

Another important question is to explore an analogue of the collapse of PH for learnability. In other words, does polynomial time learnability of NP/poly imply polynomial time learnability of PH/poly? Note that, under a strong assumption of the existence of a (possibly adaptive and non-relativisable) worst-case to average-case reduction for NP, we can use the techniques in Lemma 5.20 along with the downward-self-reducibility of SAT to show such a collapse.

On the other hand, [Imp11] also shows that there exists an oracle $\mathcal{O}$ with respect to which $\mathsf{DistNP}^{\mathcal{O}} \subseteq \mathsf{AvgP}^{\mathcal{O}}$ and $\Sigma_2^{\mathcal{O}} \not\subseteq \mathsf{HeurSIZE}^{\mathcal{O}}[2^{n^\alpha}]$. Essentially, this result negates the existence of any *relativisable reductions* which show a statement analogous to the PH collapse for average-case algorithms, i.e. if NP is easy on average, then Σ_2 is easy on average too. In a similar spirit, can we prove that no relativisable technique can show that if NP/poly is learnable in polynomial time, then Σ_2/poly is learnable in polynomial time as well?

[9] The proof of Lemma 3.6 in [CHO+20] uses a different and slightly more complex technique involving random sampling to estimate the closeness of the hypothesis, compared to the one in Section 3.3.

Chapter 6

Deterministic #SAT Algorithm for $\mathsf{AC}^0[2]$

6.1 Introduction

Our understanding of the class of general Boolean circuits is very limited. We have no super-linear lower bounds against them for any explicit function, nor do we have non-trivial solutions for any of the meta-algorithmic tasks we have endeavoured to study in this thesis. On the other hand, the situation is far better for more restricted classes of circuits, especially circuits of *constant depth* where the gates have unbounded fan-in, both with respect to lower bounds and to meta-computational questions.

In this chapter, we focus on the class $\mathsf{AC}^0[2]$, i.e. constant-depth circuits with AND, OR and XOR gates (or more generally, AND, OR and MOD_p gates for some prime p). This is a class that has been intensively studied. Razborov [Raz87] and Smolensky [Smo87] showed super-polynomial size lower bounds against this class for very simple functions such as the Majority function and the MOD_q function where q is a prime different from 2. As we discuss in Section 6.2, non-trivial randomised algorithms for deciding satisfiability of circuits from this class are implicit in [Wil14b, LPT+17]. [CIKK16] also gave a quasi-polynomial time algorithm for learning $\mathsf{AC}^0[2]$ circuits in the membership query model.

However, there are still several gaps in our understanding of $\mathsf{AC}^0[2]$. With regards to lower bounds, we still do not have *tight* lower bounds for functions like Majority or MOD_q. The Razborov-Smolensky approximation method yields size lower bounds of the form $2^{\Omega(n^{1/2(d-1)})}$ for Majority against $\mathsf{AC}^0[2]$ circuits of depth d over n variables, and a standard divide-and-conquer strategy yields an upper bound of $2^{\widetilde{O}(n^{1/(d-1)})}$. Recently, [OSS19] showed that for large constant depths, this upper bound can be improved to $2^{\widetilde{O}\left(n^{2/3(d-4)}\right)}$, and the lower bound to $2^{\Omega(n^{1/2d-4})}$. Closing this gap in the exponent between upper and lower bounds for Majority has been a long-standing open question in the complexity theory of constant-depth circuits (for depths 3 and 4, [OSS19] show tight size bounds).

Note that, in contrast, tight bounds are known up to constant factors in the exponent when only AND and OR gates are allowed - Parity is known to have complexity $2^{\Theta(n^{1/(d-1)})}$ in this simpler model [Ajt83, FSS84, Yao85, Hås86].

With regard to meta-algorithmic tasks, despite the learning breakthrough of [CIKK16] mentioned previously, the situation for pseudo-random generators (PRGs) and satisfiability algorithms is still unclear. Very recently, [CR20, CLW20] used the algorithmic method to show strong average-case lower bounds for E^{NP} against $\mathsf{AC}^0[2]$ circuits of sub-exponential size (and actually, ACC^0), and consequently E^{NP}-computable PRGs of $\mathsf{poly}(\log n)$ seed length, on almost every input length. In the case of satisfiability algorithms, randomised algorithms improving non-trivially over brute force search are implicit in previous work, but good deterministic algorithms were unknown prior to this work.

Deterministic satisfiability algorithms are important for a couple of reasons. First, they indicate an improved structural understanding of the circuit class in question, often requiring new techniques to design. Second, they imply circuit lower bounds via the connection of Williams [Wil13a] - such an implication is not known from randomised algorithms.[1]

Note that even under standard derandomisation assumptions, it is unclear how to get a non-trivial deterministic satisfiability algorithm from a non-trivial randomised satisfiability algorithm. The reason is that derandomisation inherently incurs a quadratic slowdown. The deterministic simulation of a randomised algorithm running in time T will take time at least T^2 when a PRG is used to do the derandomisation, as the range of the PRG will have size at least T. This quadratic slowdown is unaffordable in the parametric regime under consideration here - we are interested in algorithms running in time $2^{n-g(n)}$, for some $g(n) = o(n)$. Hence the determinisation procedure cannot be black-box, but must rather use refined structural information about the circuit class in question.

The main result of this chapter is a *deterministic algorithm for counting satisfying assignments* of $\mathsf{AC}^0[2]$ circuits that improves non-trivially over brute force search.

Theorem 6.1 (Main theorem)**.** *The following holds for some absolute constant $\varepsilon_0 > 0$. There is a deterministic algorithm that given an $\mathsf{AC}^0[2]$ circuit C over n variables such that C has depth at most d and size $s \leq 2^{(\varepsilon_0 n)^{1/d}}$, counts the number of satisfying assignments to C in time 2^{n-t} where $t = t(n, s, d) = n/O(\log s)^{d-1}$.*

Remark 6.2. *It is easy to generalise our results to work with $\mathsf{AC}^0[p]$ circuits for any fixed prime p.*

[1] One-sided error randomised algorithms, and more generally, co-non-deterministic algorithms for satisfiability for the circuit class also imply lower bounds via the result of Williams [Wil13a]. However, it is easy to check that the previous randomised algorithms for $\mathsf{AC}^0[2]$ were two-sided error algorithms.

In terms of parameters, the savings over brute force search matches the savings in the randomised algorithm of [IMP12] for AC^0 circuits when the circuit size is $n^{1+\Omega(1)}$. Thus any further improvement in savings in our result would give a corresponding improvement for AC^0 algorithms, and moreover via the connection of Williams [Wil13a] to circuit lower bounds, a strong improvement would give super-polynomial formula size lower bounds for non-deterministic exponential time.

Next, we use Theorem 6.1 to get better lower bounds against $\mathsf{AC}^0[2]$ circuits than known before by using the connection of Williams [Wil13a]. In fact, we need a refinement of the connection due to [BSV14]. Our lower bound holds for a language in E^{NP}.

Theorem 6.3. *For any positive integer d, there is a language in E^{NP} which does not have $\mathsf{AC}^0[2]$ circuits of depth d and size $2^{o(n^{1/(d+1)})}$.*

Remark 6.4. *In very recent work [Vio20], Viola showed an improvement to Theorem 6.3. He showed the existence of a language in E^{NP} which requires $\mathsf{AC}^0[2]$ circuits of depth d and size $2^{\Omega(n/\log^2 n)^{1/d-1}}$, for every constant d. This lower bound almost matches the $2^{\Omega(n^{1/d-1})}$ lower bounds for AC^0 via switching lemma [Hås86].*

6.2 Proof Outline for the Main Theorem

Our starting point is a *randomised* algorithm for the problem of checking satisfiability of an $\mathsf{AC}^0[2]$ circuit C that runs in time 2^{n-m} where $m = n/O(\log s)^{d-1}$, and s, d represent the size, depth of C respectively (we also assume that s is suitably upper bounded, but we ignore it in this section). This algorithm is essentially due to Williams [Wil14b] and Lokshtanov, Paturi, Tamaki, Williams and Yu [LPT+17], though it does not appear explicitly in either of these works.

The idea is to use a result of Razborov [Raz87] that essentially says that small $\mathsf{AC}^0[2]$ circuits C can be "approximated" by polynomials of low degree. More formally, there is a randomised algorithm that, when given a circuit C of size s over n variables, produces a (random) polynomial $\mathcal{P} \in \mathbb{F}_2[x_1, \ldots, x_n]$ of degree $O(\log s)^{d-1}$ that agrees with the value of a circuit C on any given input with good probability (say 0.9). Along with a fast polynomial evaluation algorithm [Wil11], this immediately yields an *enumeration* algorithm for C (i.e. an algorithm to output the truth table of C) that runs in time $\mathsf{poly}(n)2^n + \mathsf{poly}(s)$, which beats (for large enough s) the trivial algorithm that simply evaluates C on each input and hence takes time $s \cdot 2^n$. Repeating the algorithm $\mathsf{poly}(n)$ times and taking the majority vote on each input, we get an enumeration algorithm that works with high probability.

To obtain a randomised satisfiability algorithm that runs in better-than-brute-force time, we use the above idea along with the "blowup-trick" [Wil14c, CW16, LPT+17]. For any $a \in \{0,1\}^m$, let C_a be the circuit obtained by setting the last m variables of C to a. Note that the satisfiability of C can be computed by checking the satisfiability of $C' = \bigvee_{a \in \{0,1\}^m} C_a$, and C' is a circuit of larger size ($s \cdot 2^m$) but fewer variables ($n - m$). We now run the enumeration algorithm above on C' to check if it is satisfiable. Since the circuit is larger, the polynomial produced has larger degree: a careful analysis reveals the degree to be $m \cdot O(\log s)^{d-1}$. Setting $m = n/\Theta(\log s)^{d-1}$, we obtain a polynomial of degree $\ll n$, which can be computed in better-than-brute-force time. Running the enumeration algorithm as above gives the required satisfiability algorithm for C, which now runs in time $2^{n - n/\Theta(\log s)^{d-1}}$.

The above algorithm can further be modified to *count* satisfying assignments, by instead defining $C' = \sum_a C_a$ (a sum over $\mathbb{Z}$) instead of using an OR. Now, an additional idea is required since the polynomials $\mathcal{P}_a$ approximating each C_a are $\mathbb{F}_2$-polynomials whereas the sum is over the integers (in the satisfiability case above, the OR gate can further be approximated by a constant-degree polynomial using an idea of Razborov [Raz87], but this idea is not available here). What comes to our rescue is an idea of Toda [Tod91] and its subsequent quantitative refinement due to Beigel and Tarui [BT94] which tells us that we can simulate a sum (over $\mathbb{Z}$) of K many $\mathbb{F}_2$-polynomials of degree at most D as a polynomial (over $\mathbb{Z}$) of degree at most $D \log K$. Using this idea, we are able to obtain a polynomial of degree $m^2 \cdot O(\log s)^{d-1}$. Overall, this yields an algorithm with slightly worse running time $2^{n - \sqrt{n/\Theta(\log s)^{d-1}}}$.

These algorithms were partially derandomised by Chan and Williams [CW16] for the case that the $\mathsf{AC}^0[2]$ circuits are k-CNFs and generalised by Lokshtanov et al. [LPT+17] to the case of ANDs of degree-k polynomials.[2] Chan and Williams [CW16] observed that Razborov's randomised construction of polynomials could be suitably derandomised using *ε-biased spaces* [NN93] (which is nothing but the support of an ε-biased distribution). Using this idea (and more work), it was shown that the number of satisfying assignments to a set of degree k-polynomials in n variables could be computed in time $2^{n - n/\Theta(k)}$, which meets the running time of the satisfiability algorithm mentioned above in this special case.

However, it is unclear how to extend the ideas to the setting of general $\mathsf{AC}^0[2]$ circuits since these results used a very special property of the randomised polynomial construction for a single OR gate (and, dually, a single AND gate): namely, that there is a *constant-degree* polynomial whose bias perfectly predicts whether the input to an OR-gate is a 1

[2] Note that any k-clause is in particular a degree-k polynomial and hence the latter result generalises the former.

input or a 0 input.[3] Unfortunately, such a strong property is not known for general $\mathsf{AC}^0[2]$ circuits: the best we can hope for is to construct a polynomial that with high probability, say $1-\varepsilon$, equals the output of the circuit on any given input, but then we have to pay for this precision in terms of the degree of the polynomial constructed. Further, it is not clear how to derandomise this general inductive construction.

We start by derandomising the higher-depth random polynomial construction. Once again, ε-biased distributions play a crucial role, and we need to further use derandomised sampling using expanders for a near-optimal derandomisation. Using this along with the idea of Beigel and Tarui [BT94] would yield a deterministic algorithm for counting satisfying assignments in time $2^{n-\sqrt{n/\Theta(\log s)^{d-1}}}$.

However, we further improve the running time to $2^{n-n/\Theta(\log s)^{d-1}}$, matching the running time of the randomised algorithm for checking satisfiability. The principal idea here is to observe (by looking inside the Razborov construction) that the polynomials $\mathcal{P}_a$ computed for approximating the individual circuits C_a mentioned above have a very special form: each $\mathcal{P}_a$ is a majority of $\ell = O(m)$ many polynomials $\mathcal{P}_{a,1}, \ldots, \mathcal{P}_{a,\ell}$ of degree $O(\log s)^{d-1}$ each. We use this and some basic Fourier analysis of Boolean functions to write $\mathcal{P}_a$ as a real-valued sum of polynomials of degree at most $O(\log s)^{d-1}$; this idea is inspired by a recent result of Chen and Papakonstantinou [CP19], who use it to give an improved depth-reduction result for constant-depth circuits with Mod_q gates for composite q. The advantage of doing this is that the degree blow-up in the randomised #SAT algorithm outline above is restricted to applying the idea of Beigel and Tarui, which means that the degree drops to $m \cdot O(\log s)^{d-1}$. Setting m suitably, we now obtain a deterministic algorithm running in time $2^{n-n/O(\log s)^{d-1}}$.

6.3 Preliminaries

We will consider polynomials over the fields $\mathbb{R}$ and $\mathbb{F}_2$. We identify $\mathbb{F}_2$ with $\{0,1\}$ in the natural way. We use $\binom{n}{\leq k}$ to denote $\sum_{i=0}^{k}\binom{n}{i}$. All algorithms will be implemented in the standard Turing machine with RAM model.

We recall that any function $f : \{0,1\}^n \to R$, for any commutative ring R has a unique representation as a multilinear polynomial. Given two such polynomials representing possibly different functions f_1 and f_2, we can compute the multilinear polynomial

[3] Briefly, the Razborov polynomial for the OR function on input bits $x_1, \ldots, x_s$ is as follows. Choose $a_1, \ldots, a_s \in \mathbb{F}_2$ independently and uniformly at random and compute $\ell(x) = \sum_i a_i x_i$. Now, note that if $\mathsf{OR}(x_1, \ldots, x_s) = 1$, then $\ell(x)$ computes a uniformly random element of $\mathbb{F}_2$ and otherwise, $\ell(x) = 0$ with probability 1.

corresponding to their product by multiplying the polynomials f_1 and f_2 and "multilinearising" by replacing each copy of x_i^2 by x_i. In particular, this idea yields the following easy algorithm.

Fact 6.5. *Let R be either $\mathbb{F}_2$ or $\mathbb{Z}$. There is a deterministic algorithm which, when given as input multilinear polynomials $f_1, \ldots, f_t \in R[x_1, \ldots, x_n]$ (as a sum of monomials) of degree $d_1, \ldots, d_t$ such that $\sum_i d_i \leq D$, computes the multilinear polynomial corresponding to the product $f = f_1 \cdots f_t$. The algorithm runs in time $\mathsf{poly}(\binom{n}{\leq D})$ when $R = \mathbb{F}_2$, and time $\mathsf{poly}(\binom{n}{\leq D}, B)$ when $R = \mathbb{Z}$, where B is the bit-complexity of the coefficients of $f_1, \ldots, f_t$.*

6.3.1 Polynomials over $\mathbb{F}_2$ and Probabilistic polynomials

For our convenience, we recall the definition of probabilistic polynomials.

Definition 6.6 (Probabilistic Polynomials [Raz87, Smo87])**.** *A* probabilistic polynomial *from $\mathbb{F}_2[x_1, \ldots, x_n]$ is a random multilinear polynomial $\mathcal{P}$ (chosen according to some distribution) from $\mathbb{F}_2[x_1, \ldots, x_n]$. We say that $\mathcal{P}$ has degree at most D if the distribution of $\mathcal{P}$ is supported on polynomials of degree at most D (or equivalently $\Pr_{\mathcal{P}}\{\deg(\mathcal{P}) \leq D\} = 1$).*

We say that $\mathcal{P}$ is an ε-error probabilistic polynomial for a function $f : \{0,1\}^n \to \{0,1\}$ if for each $a \in \{0,1\}^n$, we have $\Pr_{\mathcal{P}}\{\mathcal{P}(a) \neq f(a)\} \leq \varepsilon$.

6.3.2 Polynomials over $\mathbb{R}$ and Modulus-amplification

Section 2.5 defines the Fourier expansion of a Boolean function over the $\{-1, 1\}$ basis. In this section we given an analogous definition over the $\{0, 1\}$ basis.

Recall the following basic facts about writing Boolean functions as multilinear polynomials over the *reals*. See, e.g. O'Donnell [O'D14] for proofs.

Fact 6.7. *Let $f : \{0,1\}^\ell \to \{0,1\}$ be any Boolean function.*

1. *f can written as a unique* real-valued *linear combination*

$$f(x) = \sum_{S \subseteq [\ell]} \alpha_S \chi_S(x)$$

where $\chi_S(x) = \bigoplus_{i \in S} x_i$ (note that we interpret $\chi_S(x) \in \{0,1\}$ as a real number and the sum above is taken over $\mathbb{R}$).

2. *For each S, $\alpha_S \in [-1, 1]$ and is moreover an integral multiple of $2^{-(\ell+1)}$.*

3. *There is a deterministic algorithm* FC *which, given as input f (via its truth table), computes all the above α_S's in time $2^{O(\ell)}$.*

We also define the Fourier l_1 norm of f as $\|f\|_1 = \sum_S |\alpha_S|$.

The following is a useful *Modulus-amplification* lemma due to Beigel and Tarui [BT94]. This particular version is from the work of Chan and Williams [CW16].

Lemma 6.8 (Beigel-Tarui [BT94])**.** *For every positive integer t, the degree $2t-1$ polynomial $F_t(y) \in \mathbb{Z}[y]$ defined by*

$$F_t(y) = 1 - (1-y)^t \sum_{j=0}^{t-1} \binom{t+j-1}{j} y^j$$

has the property that for all $b \in \mathbb{Z}$,

- *if $b \equiv 0 \pmod 2$, then $F_t(b) \equiv 0 \pmod{2^t}$, and*
- *if $b \equiv 1 \pmod 2$, then $F_t(b) \equiv 1 \pmod{2^t}$.*

Many satisfiability algorithms for circuits are based on evaluating multivariate polynomials efficiently over grids. The following lemma can be found in, e.g. [Wil11].

Lemma 6.9 (Fast Polynomial Evaluation)**.** *There is a deterministic algorithm* FPE*, which given as input a multilinear polynomial $P \in \mathbb{Z}[x_1, \ldots, x_n]$ as a sum of monomials, computes the values $(P(a))_{a \in \{0,1\}^n}$ in time $\mathsf{poly}(n, B) \cdot 2^n$ where B is an upper bound on the bit complexity of the coefficients of P.*

6.3.3 Small-biased generators

Recall the notion of an *ε-biased generators* from Definition 4.3, which are generators that ε-fool parities.

For any subspace W of $\{0,1\}^n$, define the indicator function $\mathbf{1}_W : \{0,1\}^n \to \{0,1\}$ as $\mathbf{1}_W(z) = 1$ if and only if the vector $z \in W$. From an Observation in O'Donnell's book [O'D14], we see that

Proposition 6.10 (Proposition 3.11 of [O'D14])**.** *Let W be a subspace of $\{0,1\}^n$ and $W_\perp$ be its orthogonal complement such that $dim(W_\perp) = k$. Then, the constant term in the Fourier expansion of the indicator function $\mathbf{1}_W$ is $\frac{1}{2^k}$. Moreover, $\|\mathbf{1}_W\|_1 = 1$.*

The proof for Proposition 6.10 is based on the fact that a vector $z \in W$ if and only if the dot product of z with every basis vector of $W_\perp$ is 0.

De, Etesami, Trevisan and Tulsiani [DETT10] observed that ε-biased generators also fool functions with small Fourier l_1-norm.

Lemma 6.11 (Lemma 2.5 of [DETT10]). *Let $G : \{0,1\}^r \to \{0,1\}^n$ be an ε-biased generator. For every function $f : \{0,1\}^n \to \mathbb{R}$ we have,*

$$\left|\mathbf{E}_{y\sim U_r}[f(G(y))] - \mathbf{E}_{x\sim U_n}[f(x)]\right| \leq \varepsilon\|f\|_1$$

From Proposition 6.10 and Lemma 6.11, we state the following useful corollary.

Corollary 6.12. *Let W be a subspace of $\{0,1\}^n$, such that the co-dimension of W is k and let $G : \{0,1\}^r \to \{0,1\}^n$ be an ε-biased distribution. Then*

$$\Pr_{y\sim U_r}\{G(y) \in W\} \in \left(\frac{1}{2^k} - \varepsilon, \frac{1}{2^k} + \varepsilon\right)$$

Intuitively speaking, Corollary 6.12 states that an ε-biased generator also ε-fools conjunctions of parities.

6.3.4 Expanders

Proofs of the following well-known facts may be found in the monograph of Hoory, Linial and Wigderson [HLW06].

Given an undirected Δ-regular multigraph G, we denote by $A(G)$ its adjacency matrix and $\widetilde{A}(G) = (1/\Delta)A(G)$ its *normalised* adjacency matrix. Then, we have the following.

- The all-1s vector $v \in \mathbb{R}^n$ is an eigenvector of $\widetilde{A}(G)$ with eigenvalue 1.
- G is connected if and only if v is the only such eigenvector (up to scalar multiplication).

Definition 6.13 (Expanders [HLW06]). *An undirected multigraph G is an (N, Δ, λ)-*expander *if it is a Δ-regular connected graph on N vertices (which we will identify with $[N]$), and all the eigenvalues (counted with multiplicity) of $\widetilde{A}(G)$ other than 1 are bounded by λ in absolute value.*

Let $\{G_n\}_{n\geq 1}$ be a sequence of expander graphs with G_n being an $(f(n), \Delta, \lambda)$-expander for some increasing function $f : \mathbb{N} \to \mathbb{N}$ and constants Δ and λ. We say that $\{G_n\}_{n\geq 1}$ is explicit *if there is a deterministic algorithm that, when given as input n, produces the graph G_n in time $\mathsf{poly}(n, f(n))$.*

We use Reingold, Vadhan and Wigderson's [RVW02] explicit construction for an expander graph.

Theorem 6.14 ([RVW02]). *For every fixed $\lambda > 0$, there exists a constant $\Delta > 0$ and an explicit sequence of expander graphs $\{G_n\}_{n\geq 1}$, where G_n is a $(2^n, \Delta, \lambda)$-expander graph, for some large enough constant Δ. Further, we can assume that Δ is a power of 2.*

We will need the following expander-based Chernoff bound due to Gillman [Gil98]. The version below is due to Healy [Hea08].

Theorem 6.15 ([Gil98, Hea08])**.** *Let G be an (N, D, λ)-expander graph and let $S \subseteq [N]$ be a subset of the vertices of G such that $|S| = \beta N$. Consider the natural ℓ-step random walk on G defined by choosing a uniformly random vertex $u_1 \in [N]$ and repeatedly choosing random neighbours $\ell - 1$ times to obtain a (random) sequence $(u_1, \ldots, u_\ell)$ of vertices of G. Let X_S denote the number of $i \in [\ell]$ such that $u_i \in S$. For any fixed $\rho \in (0, 1)$, we have*

$$\Pr_{u_1,\ldots,u_\ell}\{|X_S - \beta\ell| \geq \rho\ell\} \leq 2\exp\left(-\frac{1}{4}\rho^2(1-\lambda)\ell\right).$$

6.4 The #SAT algorithm

In this section we prove Theorem 6.1. We start with a derandomised version of a lemma by Razborov [Raz87] for approximating $\mathsf{AC}^0[2]$ circuits by low-degree polynomials. Using this version, we then state formally the #SAT algorithm and analyse it.

6.4.1 Derandomised construction of probabilistic polynomials for $\mathsf{AC}^0[2]$

The following lemma is an algorithmic version of a result by Razborov [Raz87] (see also Kopparty-Srinivasan [KS18] for the dependence on ε). It can be viewed as a derandomisation of Williams' [Wil14b] algorithm.

Lemma 6.16. *For any $\varepsilon > 0$, an $\mathsf{AC}^0[2]$ circuit C over n variables of depth d and size at most s has an ε-error probabilistic polynomial $\mathcal{P}$ from $\mathbb{F}_2[x_1, \ldots, x_n]$ of degree at most $D = (O(\log s)^{d-1} \cdot \log(1/\varepsilon))$. Further, $\mathcal{P} = \mathsf{Maj}(\mathcal{P}_1, \ldots, \mathcal{P}_\ell)$, where $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$ are probabilistic polynomials of degree $D_1 = O(\log s)^{d-1}$ and $\ell = O(\log(1/\varepsilon))$.*

Moreover, there is a deterministic procedure $\mathcal{S}$, which when given as input the circuit C, the parameter ε, and a uniformly random Boolean string $\boldsymbol{\sigma}$ of length $r = O(\log(s/\varepsilon))$, produces a random sample of the polynomials $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$ as sums of monomials. The procedure $\mathcal{S}$ runs in time $\mathsf{poly}\left(\ell, s, \binom{n}{\leq D_1}\right)$.

Before we go into the proof of Lemma 6.16, we prove a weaker version that will be useful for proving it. This result is a higher-depth analogue of a result of Chan and Williams [CW16] which itself may be viewed as a derandomisation of Razborov's construction [Raz87] for depth-2 circuits.

Lemma 6.17. *For every* $\mathsf{AC}^0[2]$ *circuit* C *over* n *variables of depth* d *and size* s*, there exists a* $\frac{1}{4}$*-error probabilistic polynomial* $\mathcal{P}'$ *from* $\mathbb{F}_2[x_1, \ldots, x_n]$ *of degree at most* $D_1 = O(\log s)^{d-1}$*. Further, there is a deterministic algorithm* $\mathcal{T}$ *that produces a random sample of* $\mathcal{P}'$ *as a sum of monomials given as input* C, s *and* $O(\log s)$ *random bits. The algorithm* $\mathcal{T}$ *runs in time* $\mathsf{poly}\left(s, \binom{n}{\leq D_1}\right)$.

Proof. Let C be a circuit over n variables of size s and depth d. Let $m = s\log(40s), \delta = 1/(20s)$ and G be an δ-biased generator over $\{0,1\}^m$ with seed length $O(\log(m/\delta)) = O(\log s)$ (from Lemma 4.4).

Fix some enumeration $g_1, \ldots, g_s$ of all the gates in C. Let $h \in \{g_1, \ldots, g_s\}$ be the output gate of C and let $C_1, \ldots, C_r$ ($r \leq s$) be the depth-$(d-1)$ sub-circuits feeding into h. Firstly, for each gate $g \neq h$ in the circuit, we construct a probabilistic polynomial of degree $O(\log s)$ over the gate inputs, following which we construct a constant degree probabilistic polynomial for the gate h. Composing these polynomials together gives the constant error probabilistic polynomial of degree $O(\log s)^{d-1}$ for C.

Let $x \in \{0,1\}^n$ be any input to C. Fix any gate $g \neq h$. If g is a NOT gate or a MOD_2 gate, we can easily get a polynomial P_g of degree 1 which always agrees with the function computed by g and needs no random bits. Therefore, let g be an OR gate (a dual construction works for AND gates). Let $g_{i_1}, \ldots, g_{i_k}$ be the gates that are inputs to g and $v_g \in \{0,1\}^s$ such that $v_g[i] = g_i(x)$ iff $i \in \{i_1, \ldots, i_k\}$. Observe that g outputs 1 iff v_g is not the zero vector.

Let $t = \log(40s)$. Construct the vectors $u_1, \ldots, u_t \in \{0,1\}^m$, such that for each u_p, $1 \leq p \leq t$, on dividing u_p into t blocks, the p^{th} block contains v_g and all the other bits of u_p are set to 0. The dimension of the vector space $V \subseteq \mathbb{F}_2^m$ spanned by $\{u_1, \ldots, u_t\}$ is equal to t if v_g is a non-zero vector. Use the generator G to sample a string $y \in \{0,1\}^m$, using a seed of length $O(\log s)$. Consider y as being split into t blocks $y_1, \ldots, y_t$ of size s each.

W.l.o.g. consider the vector u_1. The inner product $\langle u_1, y\rangle$ which is exactly equal to the inner product $\langle v_g, y_1\rangle$, can be calculated by hardwiring into a MOD_2 gate all the input gates of g for which the corresponding bit in y equals 1. In other words, the inner product $\langle u_1, y\rangle$ is represented by the polynomial $q_1^y = \sum_{j \in \{i_1, \ldots, i_k\}} (y_1)_j \cdot g_j(x)$ over inputs $g_{i_1}(x), \ldots, g_{i_k}(x)$. Repeating this construction for all t vectors $u_1, \ldots, u_t$, we get polynomials $q_1^y, \ldots, q_t^y$. Finally, we compute the disjunction of these t terms using the polynomial $P_g^y = 1 - \prod_{p=1}^{t}(1 - q_p^y)$ over inputs $g_{i_1}(x), \ldots, g_{i_k}(x)$, of degree $t = O(\log s)$.

We now analyse the behaviour of P_g^y on input x. If the gate g outputs 0 on x, then for any $y \in S$, we get $\langle u_p, y\rangle = 0$ for every $1 \leq p \leq t$. In particular, P_g^y outputs 0 with probability 1. On the other hand, if $g(x) = 1$, we see that P_g^y errs on x iff for each $1 \leq p \leq t$, $q_p^y = 0$. Let $V_\perp$ be the orthogonal complement to $V = \text{span}(\{u_1, \ldots, u_t\})$.

Note that the co-dimension of the vector space $V_\perp$ is t. We now use Corollary 6.12 to see that, if $g(x) = 1$, y belongs to $V_\perp$, or in other words, $\langle u_p, y\rangle = 0$ for every $1 \le p \le t$, with probability at most $\frac{1}{2^t} + \delta$. Thus, on input x, P_g^y disagrees with the output of g with probability at most $\frac{1}{2^t} + \delta = \frac{1}{40s} + \frac{1}{20s} = \frac{3}{40s}$.

We generate a *single* y from G and then use it (for each OR and AND gate) to get the probabilistic polynomial P_g^y for each gate $g \neq h$ in C. For each of the depth-$(d-1)$ sub-circuits $C_1, \ldots, C_r$, we compose the polynomials for their gates inductively, to obtain probabilistic polynomials $\mathcal{Q}_1, \ldots, \mathcal{Q}_r$ of degree at most $O(\log s)^{d-1}$. Following this, we use the first $t_1 = 3$ blocks of y in a similar fashion to get a probabilistic polynomial $\mathcal{P}_h$ of degree $O(1)$ for the function computed by the output gate h with error at most $\frac{1}{2^{t_1}} + \delta$. Now, for any input $x \in \{0,1\}^n$, $\mathcal{P}' = \mathcal{P}_h(\mathcal{Q}_1, \ldots, \mathcal{Q}_r)$ satisfies

$$\begin{aligned}
\Pr_{\mathcal{P}'}\{C(x) \neq \mathcal{P}'(x)\} &\le \Pr_{\mathcal{Q}_1,\ldots,\mathcal{Q}_r}\{\exists j \in [r] : C_j(x) \neq \mathcal{Q}_j(x)\} \\
&\qquad + \Pr_{\mathcal{P}}\{h(C_1(x), \ldots, C_r(x)) \neq \mathcal{P}_h(C_1(x), \ldots, C_r(x))\} \\
&\le s \cdot \frac{3}{40s} + \frac{1}{2^{t_1}} + \delta \\
&\le \frac{3}{40} + \frac{1}{8} + \frac{1}{20s} \le \frac{1}{4}
\end{aligned}$$

where the second inequality uses a union bound over the (at most s) gates g in C.

This gives us a probabilistic polynomial $\mathcal{P}'$ of degree $O(\log s)^{d-1}$ for the circuit C, with error at most $\frac{1}{4}$ and the number of random bits needed to get a sample is the seed length of G, which is $O(\log s)$. This polynomial is multilinear and by expanding the monomials at each step we ensure multilinearity of the intermediate polynomials. Using Fact 6.5 we see that the running time of the sampling algorithm is given by $\mathsf{poly}\left(s, \binom{n}{\le D_1}\right)$. □

Proof of Lemma 6.16. Let $k = O(\log s)$ be the number of random bits needed by the algorithm $\mathcal{T}$ from Lemma 6.17. Consider an explicit $(2^k, \Delta, \lambda)$ expander graph G on $V = \{0,1\}^k$ given by Theorem 6.14, where Δ is a large enough constant given by the construction and $\lambda = \frac{1}{2}$. The graph G can be constructed in time $\mathsf{poly}(2^k) = \mathsf{poly}(s)$.

Let $\ell = 200 \log\left(\frac{2}{\varepsilon}\right)$. We define the algorithm $\mathcal{S}$ to be the following deterministic procedure which takes as input the circuit C, the parameter ε and a uniformly random seed $\boldsymbol{\sigma}$ of length $r = k + (\ell - 1) \cdot \log \Delta$ and produces a random sample of the probabilistic polynomials $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$, where each of the $\mathcal{P}_i$ is an instantiation of the probabilistic polynomial constructed in Lemma 6.17.

1. Perform a length ℓ random walk in G using the bits of the random seed $\boldsymbol{\sigma}$ to obtain $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\ell \in V = \{0,1\}^k$, i.e. first choose $\boldsymbol{u}_1$ uniformly at random from V and for each $i \in \{2, \ldots, \ell\}$, let $\boldsymbol{u}_i$ be a random neighbour of $\boldsymbol{u}_{i-1}$ in the graph G.

2. For each $1 \leq i \leq \ell$, use $\boldsymbol{u}_i$ as the input string of random bits to algorithm $\mathcal{T}$ from Lemma 6.17 to obtain a random sample $\mathcal{P}_i$ of the 1/4-error probabilistic polynomial for C.

The number of random bits used by $\mathcal{S}$ to obtain a random sample of $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$ is $r = O(k + \ell) = O(\log s + \log\left(\frac{1}{\varepsilon}\right)) = O\left(\log(s/\varepsilon)\right)$. At each step of the random walk we spend $\mathsf{poly}\left(s, \binom{n}{\leq D_1}\right)$ time to get a sample and thus, the overall running time of $\mathcal{S}$ is $\mathsf{poly}\left(\ell, s, \binom{n}{\leq D_1}\right)$.

Now, define the probabilistic polynomial $\mathcal{P} = \mathsf{Maj}(\mathcal{P}_1, \ldots, \mathcal{P}_\ell)$. Since, $\mathsf{Majority}$ of ℓ bits is a polynomial of degree at most ℓ, the degree of $\mathcal{P}$ is $O(\log s)^{d-1} \cdot \log(1/\varepsilon)$.

To show that $\mathcal{P}$ is a probabilistic polynomial with error ε for the circuit C, fix an input $x \in \{0,1\}^n$. For any $u \in V$, let P^u be the polynomial sampled by the algorithm $\mathcal{T}$ when given u as input. Let B be the set of vertices $u \in V$ such that $P^u(x) \neq C(x)$. Note that as $\mathcal{T}$ samples a 1/4-error probabilistic polynomial for C, we must have $|B| \leq |V|/4$. Now, we have $\mathcal{P}(x) \neq C(x)$ iff a majority of the vertices on the random walk sampled by $\mathcal{S}$ belong to B. Using Theorem 6.15 we show that this event happens with a probability at most ε.

Let X_B be the random variable which denotes the number of $i \in [\ell]$ such that $i \in B$. Using Theorem 6.15 with the settings $\beta = \frac{|B|}{|V|} \leq \frac{1}{4}, \lambda = \frac{1}{2}, \rho = \frac{1}{4}$ and $\ell = 200\log(2/\varepsilon)$, we see that

$$\Pr_{u_1,\ldots,u_\ell}\left\{\left|X_B - \ell/4\right| > \ell/4\right\} \leq 2\exp\left(-\frac{1}{4}\rho^2(1-\lambda)\cdot 200\log\left(\frac{2}{\varepsilon}\right)\right) < 2\cdot 2^{-\log\left(\frac{2}{\varepsilon}\right)} < \varepsilon$$

□

6.4.2 The algorithm and its analysis

We begin by describing the #SAT algorithm $\mathcal{A}$.

Algorithm $\mathcal{A}$. The algorithm $\mathcal{A}$ has the following desired input-output behaviour.

> **Input**: An $\mathsf{AC}^0[2]$ circuit C over n variables of size at most s and depth at most d. Recall that $s \leq 2^{(\varepsilon_0 n)^{1/d}}$ for some absolute constant $\varepsilon_0 > 0$ (to be chosen below). We assume $s \geq n$.
>
> **Desired Output**: The number of satisfying assignments of C.

Notation. Let $m = \gamma n/(\log s)^{d-1}$ for a suitable absolute constant $\gamma > 0$ that will be fixed below. Let $\varepsilon = 1/2^{10m}$. For s and ε as defined above, choose $r = O(\log(s/\varepsilon))$ suitably so that the sampling algorithm $\mathcal{S}$ from Lemma 6.16 works as stated.

For each $\sigma \in \{0,1\}^r$, let $(P_1^\sigma, \ldots, P_\ell^\sigma)$ be the output of the algorithm $\mathcal{S}$ on string σ (note that the probabilistic polynomials $(\mathcal{P}_1, \ldots, \mathcal{P}_\ell)$ from Lemma 6.16 are exactly the polynomials $(P_1^{\boldsymbol{\sigma}}, \ldots, P_\ell^{\boldsymbol{\sigma}})$ for a uniformly random $\boldsymbol{\sigma}$, and hence $\mathcal{P} = \mathsf{Maj}(P_1^{\boldsymbol{\sigma}}, \ldots, P_\ell^{\boldsymbol{\sigma}})$).

Now, for a fixed $\sigma \in \{0,1\}^r$ and $c \in \{0,1\}^m$, let $P_i^{\sigma,c} \in \mathbb{F}_2[x_1, \ldots, x_{n-m}]$ be defined by $P_i^{\sigma,c} = P_i^\sigma(x_1, \ldots, x_{n-m}, c_1, \ldots, c_m)$ (i.e. the last m variables of P_i^σ are fixed to bits of c). Let $P^{\sigma,c} = \mathsf{Maj}(P_1^{\sigma,c}, \ldots, P_\ell^{\sigma,c})$.

For $S \subseteq [\ell]$, let $P_S^{\sigma,c} = \bigoplus_{i \in S} P_i^{\sigma,c}$. Let $Q_S^{\sigma,c}$ be the polynomial with integer coefficients obtained by treating the $\mathbb{F}_2$-coefficients of $P_S^{\sigma,c}$ as integers. Note that for each $b \in \{0,1\}^{n-m}$, $P_S^{\sigma,c}(b) \equiv Q_S^{\sigma,c}(b) \pmod 2$.

1. Using the algorithm FC from Fact 6.7, compute[4] integers $k_S \in \{2^{-(\ell+1)}, \ldots, 2^{(\ell+1)}\}$ for each $S \subseteq [\ell]$ such that $\mathsf{Maj}(z_1, \ldots, z_\ell) = \frac{1}{2^{\ell+1}} \sum_{S \subseteq [\ell]} k_S \bigoplus_{i \in S} z_i$.

2. For each $c \in \{0,1\}^m$, $\sigma \in \{0,1\}^r$ and $S \subseteq [\ell]$, construct (as a sum of monomials) the polynomial $Q_S^{\sigma,c}(x_1, \ldots, x_{n-m})$ using the algorithm $\mathcal{S}$ from Lemma 6.16.

3. Construct as a sum of monomials, the multilinear polynomial $R \in \mathbb{Z}[x_1, \ldots, x_{n-m}]$ (Fact 6.5) defined by

$$R(x_1, \ldots, x_{n-m}) = \sum_{c \in \{0,1\}^m} \sum_{\sigma \in \{0,1\}^r} \sum_{S \subseteq [\ell]} k_S \cdot F_t(Q_S^{\sigma,c}(x_1, \ldots, x_{n-m}))$$

   where F_t is the modulus amplifying polynomial given by Lemma 6.8 and for a large absolute constant $A > 0$ chosen below, $t = A \log(s/\varepsilon)$.

4. Evaluate $R(b)$ for each $b \in \{0,1\}^{n-m}$ using the algorithm FPE from Lemma 6.9.

5. Let $R_t(b) = R(b) \pmod{2^t} \in \{0, \ldots, 2^t - 1\}$. Output $\sum_{b \in \{0,1\}^{n-m}} [R_t(b)/2^{\ell+1+r}]$ where $[x]$ denotes the integer closest to x (if x is a half-integer, $[x]$ is defined arbitrarily).

Theorem 6.1 follows directly from Lemmas 6.18 and 6.19 below.

Lemma 6.18 (Running time)**.** *For any constant $A > 0$, there exist constants $\gamma > 0$ and $\varepsilon_0 > 0$ such that the algorithm $\mathcal{A}$, on an input circuit C of depth at most d and size at most $s \leq 2^{(\varepsilon_0 n)^{1/d}}$, has running time $\mathsf{poly}(s) \cdot 2^{n/10} + \mathsf{poly}(n) \cdot 2^{n-m}$.*

[4] There is actually an explicit description of the integers k_S (see, e.g., O'Donnell [O'D14]) using which each k_S can each be computed in time $\mathsf{poly}(\ell)$ as opposed to the $2^{O(\ell)}$ time taken by the algorithm C. However, the algorithm we give here doesn't need this and works for any Boolean function in place of Maj.

Proof. We analyse the running time of $\mathcal{A}$ by looking at the running times for each of its individual steps. From Fact 6.7, we see that Step 1 of the algorithm takes $2^{O(\ell)}$ running time. Since $\ell = O(\log(1/\varepsilon)) = O(m) = O(n/(\log s)^{d-1}) = o(n)$, this step takes at most $2^{o(n)} < 2^{n/10}$ time to run.

For Step 2, we see that the running time is $2^{\ell+m+r} \cdot \mathsf{poly}\left(\ell, s, \binom{n}{\leq D_1}\right)$, as we construct $2^{\ell+m+r}$ many polynomials using the algorithm $\mathcal{S}$, each of which takes $\mathsf{poly}(\ell, s, \binom{n}{\leq D_1})$ time to construct, as seen in Lemma 6.16. For the parameters we pick, we see that $2^{\ell+m+r} = 2^{o(n)}$ and $\binom{n}{\leq D_1} \leq n^{D_1} = n^{O(\log s)^{d-1}} = 2^{O(n^{(d-1)/d} \log n)}$. Thus, Step 2 takes at most $2^{n/10} \cdot \mathsf{poly}(n, s)$ time.

Step 3 takes time $2^{\ell+m+r} \cdot \mathsf{poly}\left(\ell, \binom{n}{\leq 2tD_1}\right)$, as the modulus amplifying polynomial F_t blows the degree of the polynomial up by a factor of $(2t-1)$ and the number of monomials in the multilinear expansion of the polynomial $R(x_1, \ldots, x_{n-m})$ is $\mathsf{poly}\left(\ell, \binom{n}{\leq 2tD_1}\right)$. To upper bound this running time, let $c' > 0$ be a constant such that the degree parameter D_1 from Lemma 6.16 is at most $c' \cdot (\log s)^{d-1}$. Then the degree of the polynomial R is at most

$$\begin{aligned} 2tD_1 &\leq 2t \cdot c'(\log s)^{d-1} \\ &= 2A \log(s/\varepsilon) \cdot c'(\log s)^{d-1} \\ &= 2Ac'(\log s)^d + 20Amc'(\log s)^{d-1} \\ &= 2Ac'(\log s)^d + 20A\gamma c'n \end{aligned}$$

where we have used $\log(1/\varepsilon) = 10m$ and $m = \gamma n/(\log s)^{d-1}$.

Fix the constants $\varepsilon_0 = \left(\frac{1}{400Ac'}\right)$ and $\gamma = \frac{1}{4000Ac'}$. This ensures that the $2tD_1 \leq 0.01n$. From this we see that, $\binom{n}{\leq 2tD_1}$ is at most $\left(\frac{ne}{2tD_1}\right)^{2tD_1} \leq \left(\frac{ne}{0.01n}\right)^{0.01n} < 2^{0.09n}$ and we see that step 3 takes at most $2^{n/10}$ time.

Note that each k_S computed in Step 1 is an $(\ell+1)$-bit integer and hence, the bit complexity of the coefficients of R is at most $O(\ell+m+r) \leq n$. From Lemma 6.9, we see that Step 4 takes $2^{n-m}\mathsf{poly}(n)$ time and Step 5 runs in the same time trivially. Thus, the algorithm $\mathcal{A}$ takes a total of $\mathsf{poly}(n, s) \cdot 2^{n/10} + \mathsf{poly}(n) \cdot (2^{n-m})$ to run. □

Lemma 6.19 (Correctness)**.** *Assume that $A > 0$ is chosen large enough so that $t > m+r+\ell+10$. Then the algorithm $\mathcal{A}$ above outputs the number of satisfying assignments of C.*

Proof. We need to show that the algorithm $\mathcal{A}$ computes correctly the number of satisfying assignments of C. To this end, define the function C' on $n-m$ input bits by

$$C'(x_1, \ldots, x_{n-m}) = \sum_{c \in \{0,1\}^m} C(x_1, \ldots, x_{n-m}, c_1, \ldots, c_m)$$

where the sum is over $\mathbb{Z}$. It suffices to show that, for every $b \in \{0,1\}^{n-m}$, $[R_t(b)/2^{\ell+1+r}]$ is a correct estimate of $C'(b)$ since this implies that the number of satisfying assignments of C, which is equal to $\sum_{b\in\{0,1\}^{n-m}} C'(b)$, is computed correctly.

From the definition of $R_t(b)$, we see that for any $b \in \{0,1\}^{n-m}$

$$R_t(b) = R(b) \pmod{2^t} = \left(\sum_{c\in\{0,1\}^m} \sum_{\sigma\in\{0,1\}^r} \sum_{S\subseteq[\ell]} k_S \cdot F_t(Q_S^{\sigma,c}(b)) \right) \pmod{2^t}$$
$$= \left(\sum_{c\in\{0,1\}^m} \sum_{\sigma\in\{0,1\}^r} \sum_{S\subseteq[\ell]} k_S \cdot \left(F_t(Q_S^{\sigma,c}(b)) \pmod{2^t}\right) \right) \pmod{2^t}$$

For every $\sigma \in \{0,1\}^r, c \in \{0,1\}^m, S \subseteq [\ell]$ and $b \in \{0,1\}^{n-m}$, we use the property of the modulus amplifying polynomial F_t from Lemma 6.8, to observe that $F_t(Q_S^{\sigma,c}(b)) \pmod{2^t} = Q_S^{\sigma,c}(b) \pmod 2 = P_S^{\sigma,c}(b) = \oplus_{i\in S} P_i^{\sigma,c}(b)$. This observation, along with Step 1 of the algorithm $\mathcal{A}$ implies that the sum $\sum_{S\subseteq[\ell]} k_S \cdot (F_t(Q_S^{\sigma,c}(b)) \pmod{2^t})$ is the same as the value of $2^{\ell+1} \cdot \mathsf{Maj}(P_1^{\sigma,c}(b), \ldots, P_\ell^{\sigma,c}(b)) = 2^{\ell+1} \cdot P^{\sigma,c}(b)$. In other words,

$$R_t(b) = \left(2^{\ell+1} \sum_{c\in\{0,1\}^m} \sum_{\sigma\in\{0,1\}^r} P^{\sigma,c}(b) \right) \pmod{2^t} = 2^{\ell+1} \sum_{c\in\{0,1\}^m} \sum_{\sigma\in\{0,1\}^r} P^{\sigma,c}(b)$$

where the last equality follows from the fact that $t > m + \ell + r + 10$.

Now, for every $b \in \{0,1\}^{n-m}$ and $c \in \{0,1\}^m$, we have

$$\sum_{\sigma\in\{0,1\}^r} P^{\sigma,c}(b) \begin{cases} \geq 2^r(1-\varepsilon) & \text{if } C(b,c) = 1 \\ \leq 2^r\varepsilon & \text{if } C(b,c) = 0 \end{cases}$$

where $\varepsilon = 2^{-10m}$. Since $C'(b) = \sum_{c\in\{0,1\}^m} C(b,c)$, we have that for every $b \in \{0,1\}^{n-m}$

$$2^{\ell+1} \cdot 2^r(1-\varepsilon)C'(b) \leq R_t(b), \text{ and}$$
$$R_t(b) \leq 2^{\ell+1} \cdot (2^r(1-\varepsilon)C'(b) + \varepsilon(2^m - C'(b)) \cdot 2^r)$$
$$\leq 2^{\ell+1} \cdot (2^r(1-\varepsilon)C'(b) + \varepsilon 2^m \cdot 2^r)\,.$$

In particular, since $\varepsilon = 2^{-10m}$, for every $b \in \{0,1\}^{n-m}$, we see that the estimate returned by the algorithm, which is $[R_t(b)/2^{\ell+1+r}]$, is equal to $C'(b)$. □

6.4.3 A Consequence for Lower Bounds

Our #SAT algorithm can be used to obtain improved lower bounds against $\mathsf{AC}^0[2]$ circuits (and more generally, against $\mathsf{AC}^0[p]$ circuits for prime p), using Williams' connection

between algorithms and lower bounds. These lower bounds are, however, not very explicit - they hold for a language in E^{NP}.

We first remind the reader of the best "explicit" lower bounds that are known against $\mathsf{AC}^0[2]$ circuits.

Theorem 6.20. *[OSS19] For each positive integer $d \geq 3$, there is a language computable in polynomial time that requires depth-d $\mathsf{AC}^0[2]$ circuits of size $2^{\Omega(n^{1/2d-4})}$.*

In fact, the lower bounds of Theorem 6.20 holds for Majority, for all $d \geq 3$. Note that, the earlier best known lower bound for Majority was $2^{\Omega(n^{1/2d-2})}$ by [Raz87, Smo87], for any $d \geq 2$.

In terms of parameters, the bound in Theorem 6.20 is weaker than the best known bound for AC^0 circuits as a function of d. Parity is known to require depth-d circuits of size $2^{\Omega(n^{1/(d-1)})}$. It has been a longstanding open problem to improve the lower bound in Theorem 6.20 *for some function in* E^{NP}, to match the AC^0 lower bound for Parity. Note that [OSS19] show that we can't hope to prove such a lower bound for any symmetric function (like Majority). Using the algorithmic method of Williams and its refinements, we are able to use our #SAT algorithm to make progress on this problem.

The following lemma can be shown using the proof technique of Theorem 1.5 in [BSV14].

Lemma 6.21. *[Wil13a, BSV14] Let s be a size function and d be a positive integer such that satisfiability can be solved deterministically in time $2^n/n^{\omega(1)}$ on $\mathsf{AC}^0[2]$-circuits of size $O(s(n))$ and depth at most d on n variables. Then there is a language in E^{NP} which does not have $\mathsf{AC}^0[2]$ circuits of depth $d-1$ and size $o(s(n))$.*

We now apply the lemma to get better lower bounds than Theorem 6.20 in terms of size against $\mathsf{AC}^0[2]$ circuits when the depth is at least 3. A similar lower bound against $\mathsf{AC}^0[p]$ circuits can be shown for prime p using the analogue of Theorem 6.1 for $\mathsf{AC}^0[p]$ circuits. The following result is simply a re-statement of Theorem 6.3.

Theorem 6.22. *For any positive integer d, there is a language in E^{NP} which does not have $\mathsf{AC}^0[2]$ circuits of depth d and size $2^{o(n^{1/(d+1)})}$.*

Proof. Pick $\varepsilon < \varepsilon_0$, where ε_0 is the constant in Theorem 6.1. By Theorem 6.1, for any size $s \leq 2^{(\varepsilon n)^{1/(d+1)}}$, there is a deterministic algorithm solving satisfiability of $\mathsf{AC}^0[2]$ circuits of size at most s and depth at most $d+1$ in time $2^n/n^{\omega(1)}$. Now using Lemma 6.21, we have that there is a language in E^{NP} which does not have $\mathsf{AC}^0[2]$ circuits of depth d and size $o(s(n))$, which establishes our claim. □

6.5 Future Directions

In this chapter, we showed a deterministic algorithm to count the number of satisfying assignments for $\mathsf{AC}^0[2]$-circuits of depth d and size $s \leq 2^{O(n^{1/d})}$, which beats brute-force search with savings $\frac{n}{O(\log^{d-1} s)}$. A minor caveat is that we require the circuit size to be $2^{O(n^{1/d})}$ in Theorem 6.1, for technical reasons. One would expect that the analysis can be extended to circuit size up to $2^{n^{1/(d-1)}}$.

We also use the algorithm in Theorem 6.1 to show a lower bound for E^{NP} against $\mathsf{AC}^0[2]$ circuits of depth d and size $2^{\Omega(n^{1/d+1})}$. This lower bound has recently been improved to $2^{\Omega((n/\log^2 n)^{1/d-1})}$ by [Vio20]. It would be interesting to see if we can get the $2^{\Omega(n^{1/d-1})}$ lower bound for E^{NP}, which is its best known lower bound against AC^0.

For explicit functions like $\mathsf{Majority}$, [OSS19] showed an improved $\mathsf{AC}^0[2]$-circuit size upper bound of $2^{\widetilde{O}(n^{1/(1.5d-6)})}$ (for $d \geq 5$) and a lower bound of $2^{\Omega(n^{1/(2d-4)})}$, improving on a longstanding gap between the two. An interesting question would be to find the optimal size bound for $\mathsf{Majority}$ at large depths ([OSS19] show tight bounds when the depth is 3 or 4). Another question is to use the more refined structural information about $\mathsf{AC}^0[2]$ circuits exploited in the proof of Theorem 6.1 to prove lower bounds for other explicit problems.

Additionally, we can improve our understanding of the structure of $\mathsf{AC}^0[2]$ circuits by studying other related meta-computational questions. [CIKK16] show quasi-polynomial time learning algorithms for $\mathsf{AC}^0[2]$ using membership queries over the uniform distribution. As a next step, designing learning algorithms for $\mathsf{AC}^0[2]$ using random examples, even over the uniform distribution is of great interest.

Constructing PRGs which fool $\mathsf{AC}^0[2]$ circuits is another interesting research direction. We refer to [CR20, CLW20] for recent advances on strong average-case lower bounds for $\mathsf{AC}^0[2]$, which are tightly connected to the construction of such PRGs. In particular, [CLW20] show that there exists an $\varepsilon = \varepsilon(d) > 0$ and a function in E^{NP} that cannot be $(1/2 + 1/2^{n^\varepsilon})$-approximated by any ACC^0_d circuit of size 2^{n^ε} (which contains $\mathsf{AC}^0[2]$).[5] An important feature of this result is that the lower bound holds for *almost all input lengths*, whereas previously, the algorithmic method only provided infinitely often lower bounds, including Theorem 6.22. Using this average-case lower bound, they show the construction of an E^{NP} computable PRG of seed length $\mathsf{poly}(\log n)$ which fools sub-exponential size ACC^0 circuits, on almost every input length. The questions of proving strong average-case lower bounds for functions in E, or constructing EXP-computable PRGs of seed length $n^{o(1)}$ (for stretch n) still remains a challenge.

[5] Using the techniques of [CP19], $\varepsilon = 1/O(d)$.

Chapter 7

Concluding Remarks

The results presented in this thesis explore the complexity of meta-computational problems, and encompass the study of lower bounds for the minimum circuit size problem (and minimum time bounded Kolmogorov complexity problem), hardness of learning circuit classes, #SAT algorithmic upper bounds for restricted circuit classes, and their connections with lower bounds in complexity theory. The proofs combine and extend techniques from different areas in theoretical computer science, building on approaches to circuit lower bounds developed over the last few years. Main conceptual contributions include the introduction of the locality barrier for hardness magnification, and the initiation of a top-down approach for proving hardness of learning powerful circuit classes. We also discuss future research directions at the end of each chapter. Particularly interesting directions include:

- Does the study of MCSP lead us to strong circuit lower bounds? Can we use our understanding of the locality barrier to design new lower bound techniques for MCSP or hardness magnification theorems, and prove strong separations like $\mathsf{NP} \not\subseteq \mathsf{NC}^1$?
- Is it possible to prove the NP-Hardness of PAC-learning NP/poly efficiently using random examples, via non-black-box and/or relativisable reductions? It would be interesting to even study the existence of a relativisation barrier for proving such hardness results, or for that matter, NP-Hardness of learning P/poly.
- Can we design deterministically computable PRGs of small seed length (such as $n^{o(1)}$), or learning algorithms using random examples over the uniform distribution for $\mathsf{AC}^0[2]$? Even answering these questions for slightly weaker classes like AC^0-XOR would be interesting, considering its benefits to hardness magnification.

Going forward, the main challenge for researchers is still to understand the class of general circuits better. This includes proving unconditional results on circuit lower bounds, as well as computational questions associated with related meta-computational

problems, like the hardness of learning $\mathsf{P/poly}$, uniform or non-uniform hardness of MCSP, or designing Circuit-SAT algorithms, to name a few. At this stage, we seem to need to make significant progress to tackle any of these questions, and it is unclear what sort of mathematical ideas we require for this task.

On the positive side, we have seen many recent advances in the theory of lower bounds based on its connections with meta-computational problems, and vice versa. We believe that further exploration along these lines will lead us to a more rigorous understanding of the power and limitations of various computational devices, and complexity lower bounds.

Bibliography

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABK+06] Eric Allender, Harry Buhrman, Michal Kouckỳ, Dieter Van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35(6):1467–1493, 2006.

[ABX08] Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Symposium on Foundations of Computer Science* (FOCS), pages 211–220, 2008.

[ACW16] Josh Alman, Timothy M. Chan, and R. Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 467–476. IEEE Computer Society, 2016.

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 701–710. ACM, 2006.

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[AHK17] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The minimum oracle circuit size problem. *computational complexity*, 26(2):469–496, 2017.

[AHM+08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and AC^0 Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008.

[AJ08] Alexander E. Andreev and Stasys Jukna. Very large cliques are easy to detect. *Discrete Mathematics*, 308(16):3717–3721, 2008.

[Ajt83] Miklós Ajtai. $\sum_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 1–15. Springer, 2001.

[All17] Eric Allender. The complexity of complexity. In *Computability and Complexity*, pages 79–94. Springer, 2017.

[All20] Eric Allender. The new complexity landscape around circuit minimization. In *International Conference on Language and Automata Theory and Applications*, pages 3–16. Springer, 2020.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[Ang88] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[Baz09] Louay M. J. Bazzi. Polylogarithmic independence can fool DNF formulas. *SIAM J. Comput.*, 38(6):2220–2272, 2009.

[BB15] Andrej Bogdanov and Christina Brzuska. On basing size-verifiable one-way functions on NP-hardness. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015.

[BFKL93] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Annual International Cryptology Conference*, pages 278–291. Springer, 1993.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BFT98] Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings. Thirteenth Annual IEEE Conference on Computational Complexity (Formerly: Structure in Complexity Theory Conference)(Cat. No. 98CB36247)*, pages 8–12. IEEE, 1998.

[BIS12] Paul Beame, Russell Impagliazzo, and Srikanth Srinivasan. Approximating AC^0 by small height decision trees and a deterministic algorithm for $\#\mathsf{AC}^0$-SAT. In *2012 IEEE 27th Conference on Computational Complexity*, pages 117–125. IEEE, 2012.

[BSV14] Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *International Colloquium on Automata, Languages, and Programming*, pages 163–173. Springer, 2014.

[BT94] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.

[BT06] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

[CGJ+18] Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $\mathsf{AC}^0 \circ \mathsf{MOD}_2$ lower bounds for the Boolean Inner Product. *J. Comput. Syst. Sci.*, 97:45–59, 2018.

[Che15] Ruiwen Chen. Satisfiability algorithms and lower bounds for Boolean formulas over finite bases. In *International Symposium on Mathematical Foundations of Computer Science*, pages 223–234. Springer, 2015.

[Che19] Lijie Chen. Non-deterministic quasi-polynomial time is average-case hard for ACC circuits. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1281–1304. IEEE, 2019.

[CHMY20] Mahdi Cheraghchi, Shuichi Hirahara, Dimitrios Myrisiotis, and Yuichi Yoshida. One-tape turing machine and branching program lower bounds for MCSP. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 103, 2020.

[CHO+20] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 70:1–70:48. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

[CILM18] Marco L Carmosino, Russell Impagliazzo, Shachar Lovett, and Ivan Mihajlin. Hardness amplification for non-commutative arithmetic circuits. In *33rd Computational Complexity Conference (CCC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[CJW19] Lijie Chen, Ce Jin, and Ryan Williams. Hardness magnification for all sparse NP languages. In *Symposium on Foundations of Computer Science* (FOCS), 2019.

[CJW20] Lijie Chen, Ce Jin, and R. Ryan Williams. Sharp threshold results for computational complexity. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1335–1348. ACM, 2020.

[CK15] Ruiwen Chen and Valentine Kabanets. Correlation bounds and #SAT algorithms for small linear-size circuits. In *International Computing and Combinatorics Conference*, pages 211–222. Springer, 2015.

[CKK+15] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015.

[CKLM19] Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 39:1–39:14, 2019.

[CKS16] Ruiwen Chen, Valentine Kabanets, and Nitin Saurabh. An improved deterministic #SAT algorithm for small De Morgan formulas. *Algorithmica*, 76(1):68–87, 2016.

[CLW20] Lijie Chen, Xin Lyu, and Ryan Williams. Almost-everywhere circuit lower bounds from non-trivial derandomization. In *Symposium on Foundations of Computer Science* (FOCS), 2020.

[CMMW19] Lijie Chen, Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Relations and equivalences between circuit lower bounds and Karp-Lipton theorems. In *Computational Complexity Conference* (CCC), 2019.

[COS17] Xi Chen, Igor Carboni Oliveira, and Rocco A. Servedio. Addition is exponentially harder than counting for shallow monotone circuits. In *Symposium on Theory of Computing* (STOC), pages 1232–1245, 2017.

[COS18] Ruiwen Chen, Igor C Oliveira, and Rahul Santhanam. An average-case lower bound against ACC^0. In *Latin American Symposium on Theoretical Informatics*, pages 317–330. Springer, 2018.

[CP19] Shiteng Chen and Periklis A Papakonstantinou. Depth reduction for composites. *SIAM Journal on Computing*, 48(2):668–686, 2019.

[CR20] Lijie Chen and Hanlin Ren. Strong average-case lower bounds from non-trivial derandomization. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1327–1334, 2020.

[CS12] Arkadev Chattopadhyay and Rahul Santhanam. Lower bounds on interactive compressibility by constant-depth circuits. In *Symposium on Foundations of Computer Science* (FOCS), pages 619–628, 2012.

[CS15] Ruiwen Chen and Rahul Santhanam. Improved algorithms for sparse MAX-SAT and MAX-k-CSP. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 33–45. Springer, 2015.

[CT19] Lijie Chen and Roei Tell. Bootstrapping results for threshold circuits "just beyond" known lower bounds. In *Symposium on Theory of Computing* (STOC), 2019.

[CW16] Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255. SIAM, 2016.

[CW19] Lijie Chen and R. Ryan Williams. Stronger connections between circuit analysis and circuit lower bounds, via PCPs of proximity. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 19:1–19:43. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[DETT10] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *Approximation, randomization, and combinatorial optimization. Algorithms and techniques*, pages 504–517. Springer, 2010.

[DF13] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.

[FK09] Lance Fortnow and Adam R Klivans. Efficient learning algorithms yield circuit lower bounds. *Journal of Computer and System Sciences*, 75(1):27–36, 2009.

[For02] Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. Syst. Sci.*, 65(4):612–625, 2002.

[FRS94] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multiprover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

[FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[GGM86] O Goldreich, S Goldwasser, and S Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33(4):792–807, 1986.

[GII+19] Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. $\mathsf{AC}^0[p]$ lower bounds against MCSP via the coin problem. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[Gil98] David Gillman. A Chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

[GKRS19] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 38:1–38:19, 2019.

[GKST16] Alexander Golovnev, Alexander S Kulikov, Alexander V Smal, and Suguru Tamaki. Circuit size lower bounds and $\#\mathsf{SAT}$ upper bounds through a general framework. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[GL94] Craig Gotsman and Nathan Linial. Spectral properties of threshold functions. *Combinatorica*, 14(1):35–50, 1994.

[GS10] Parikshit Gopalan and Rocco A. Servedio. Learning and lower bounds for AC^0 with threshold gates. In *International Conference on Randomization and Computation* (RANDOM), pages 588–601, 2010.

[GST07] 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. *Comput. Complex.*, 16(4):412–441, 2007.

[GV08] Dan Gutfreund and Salil P. Vadhan. Limitations of hardness vs. randomness under uniform reductions. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and*

Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings, volume 5171 of *Lecture Notes in Computer Science*, pages 469–482. Springer, 2008.

[Hås86] John Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20, 1986.

[Hås98] Johan Håstad. The shrinkage exponent of De Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

[Hea08] Alexander D Healy. Randomness-efficient sampling within NC. *Computational Complexity*, 17(1):3–37, 2008.

[HH13] Ryan C Harkins and John M Hitchcock. Exact learning algorithms, betting games, and circuit lower bounds. *ACM Transactions on Computation Theory (TOCT)*, 5(4):1–11, 2013.

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

[Hir20a] Shuichi Hirahara. Characterizing average-case complexity of ph by worst-case meta-complexity. In *Symposium on Foundations of Computer Science* (FOCS), 2020.

[Hir20b] Shuichi Hirahara. Non-disjoint promise problems from meta-computational view of pseudorandom generator constructions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 20:1–20:47. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[HLW06] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S.)*, 43(4):439–561, 2006.

[HOS18] Shuichi Hirahara, Igor C Oliveira, and Rahul Santhanam. NP-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *Proceedings of the 33rd Computational Complexity Conference*, pages 1–31, 2018.

[HP15] John M Hitchcock and Aduri Pavan. On the NP-completeness of the minimum circuit size problem. In *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[HS17] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In *Computational Complexity Conference* (CCC), pages 7:1–7:20, 2017.

[HW19] Shuichi Hirahara and Osamu Watanabe. On nonadaptive reductions to the set of random strings and its dense subsets. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:25, 2019.

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The power of natural properties as oracles. In *33rd Computational Complexity Conference (CCC 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[IL90] Russell Impagliazzo and Levin LA. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 812–821. IEEE, 1990.

[Ila20a] Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and $\mathsf{AC}^0[p]$. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[Ila20b] Rahul Ilango. The constant depth formula and partial function versions of MCSP are hard. In *Symposium on Foundations of Computer Science* (FOCS), 2020.

[ILO20] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[Imp11] Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 104–114. IEEE Computer Society, 2011.

[IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 961–972. SIAM, 2012.

[IMZ19] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):11:1–11:16, 2019.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size-depth tradeoffs for threshold circuits. *SIAM J. Comput.*, 26(3):693–707, 1997.

[IPZ01] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[IW97] Russell Impagliazzo and Avi Wigderson. $\mathsf{P} = \mathsf{BPP}$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220–229, 1997.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

[Jac97] Jeffrey C Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 1997.

[Jer09] Emil Jerábek. Approximate counting by hashing in bounded arithmetic. *J. Symb. Log.*, 74(3):829–860, 2009.

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*. Springer, 2012.

[Jus72] Jørn Justesen. Class of constructive asymptotically good algebraic codes. *IEEE Trans. Information Theory*, 18(5):652–656, 1972.

[KC00] Valentine Kabanets and Jin-Yi Cai. Circuit minimization problem. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 73–79, 2000.

[KKL+20] Valentine Kabanets, Sajin Koroth, Zhenjian Lu, Dimitrios Myrisiotis, and Igor Oliveira. Algorithms and lower bounds for De Morgan formulas of low-communication leaf gates. In *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[KKO13] Adam Klivans, Pravesh Kothari, and Igor Carboni Oliveira. Constructing hard functions using learning algorithms. In *Conference on Computational Complexity* (CCC), pages 86–97. IEEE, 2013.

[KL80] Richard M Karp and Richard J Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309, 1980.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KR13] Ilan Komargodski and Raz Ran. Average-case lower bounds for formula size. In *Symposium on Theory of Computing* (STOC), 2013.

[KRT17] Ilan Komargodski, Ran Raz, and Avishay Tal. Improved average-case lower bounds for De Morgan formula size: Matching worst-case lower bound. *SIAM Journal on Computing*, 46(1):37–57, 2017.

[KS18] Swastik Kopparty and Srikanth Srinivasan. Certifying polynomials for $\mathsf{AC}^0[\oplus]$ circuits, with applications to lower bounds and circuit compression. *Theory of Computing*, 14(1):1–24, 2018.

[KV94a] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[KV94b] Michael J Kearns and Umesh Vazirani. *An introduction to computational learning theory*. 1994.

[KW90] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM J. Discrete Math.*, 3(2):255–265, 1990.

[Lev84] Leonid A Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 27, page 52, 2020.

[LPT+17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2190–2202. SIAM, 2017.

[LW13] Richard J. Lipton and Ryan Williams. Amplifying circuit lower bounds against polynomial time, with applications. *Computational Complexity*, 22(2):311–343, 2013.

[LY94] 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.

[MMW19] Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Symposium on Theory of Computing* (STOC), 2019.

[MP20] Moritz Müller and Ján Pich. Feasibly constructive proofs of succinct weak circuit lower bounds. *Annals of Pure and Applied Logic*, 171(2):102735, 2020.

[MW17] Cody D. Murray and R. Ryan Williams. On the (non) NP-hardness of computing circuit complexity. *Theory Comput.*, 13(1):1–22, 2017.

[MW18] Cody Murray and Ryan Williams. Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 890–901, 2018.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM journal on computing*, 22(4):838–856, 1993.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.

[O'D14] Ryan O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014.

[Oli13] Igor C Oliveira. Algorithms versus circuit lower bounds. *arXiv preprint arXiv:1309.0249*, 2013.

[Oli19] Igor Carboni Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019.

[OPS19] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Computational Complexity Conference* (CCC), 2019.

[OS15] Igor Carboni Oliveira and Rahul Santhanam. Majority is incompressible by $\mathsf{AC}^0[p]$ circuits. In *Conference on Computational Complexity* (CCC), pages 124–157, 2015.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

[OS18a] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *Symposium on Foundations of Computer Science* (FOCS), pages 65–76, 2018.

[OS18b] Igor Carboni Oliveira and Rahul Santhanam. Pseudo-derandomizing learning and approximation. In *International Conference on Randomization and Computation* (RANDOM), pages 55:1–55:19, 2018.

[OSS19] Igor Carboni Oliveira, Rahul Santhanam, and Srikanth Srinivasan. Parity helps to compute majority. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[PPZ97] Ramamohan Paturi, Pavel Pudlák, and Francis Zane. Satisfiability coding lemma. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 566–574. IEEE, 1997.

[PS19] Jan Pich and Rahul Santhanam. Why are proof complexity lower bounds hard? In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1305–1324. IEEE, 2019.

[Raz85] Alexander A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Doklady Akademii Nauk SSSR*, 281:798–801, 1985. English translation in: Soviet Mathematics Doklady 31:354–357, 1985.

[Raz87] Alexander A. Razborov. Lower bounds on the size of constant-depth networks over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987.

[Rei11] Ben W. Reichardt. Reflections for quantum query algorithms. In *Symposium on Discrete Algorithms* (SODA), pages 560–569, 2011.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[RS21] Ninad Rajgopal and Rahul Santhanam. On the structure of learnability beyond P/Poly. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 46:1–46:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[RSS18] Ninad Rajgopal, Rahul Santhanam, and Srikanth Srinivasan. Deterministically counting satisfying assignments for constant-depth circuits with parity gates, with implications for lower bounds. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 78:1–78:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[RVW02] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. *Annals of Mathematics*, 155(1):157–187, 2002.

[San10] Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and qbf satisfiability. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 183–192. IEEE, 2010.

[San13] Rahul Santhanam. Ironic complicity: Satisfiability algorithms and circuit lower bounds. *Bulletin of EATCS*, 1(106), 2013.

[San20] Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Sha49] Claude E Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Symposium on Theory of Computing* (STOC), pages 77–82, 1987.

[Sri03] Aravind Srinivasan. On the approximability of clique and related maximization problems. *J. Comput. Syst. Sci.*, 67(3):633–651, 2003.

[SS96] Michael Sipser and Daniel A. Spielman. Expander codes. *IEEE Trans. Information Theory*, 42(6):1710–1722, 1996.

[SS20] Michael Saks and Rahul Santhanam. Circuit lower bounds from NP-hardness of MCSP under turing reductions. In *35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[ST13] Kazuhisa Seto and Suguru Tamaki. A satisfiability algorithm and average-case hardness for formulas over the full binary basis. *Computational Complexity*, 22(2):245–274, 2013.

[ST17] Rocco A Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[Tal14] Avishay Tal. Shrinkage of De Morgan formulae by spectral techniques. In *Symposium on Foundations of Computer Science* (FOCS), pages 551–560, 2014.

[Tal15] Avishay Tal. #SAT algorithms from shrinkage. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22, page 114, 2015.

[Tal17a] Avishay Tal. Formula lower bounds via the quantum method. In *Symposium on Theory of Computing* (STOC), pages 1256–1268, 2017.

[Tal17b] Avishay Tal. Tight Bounds on the Fourier Spectrum of AC^0. In *Computational Complexity Conference* (CCC), pages 15:1–15:31, 2017.

[Tam16] Suguru Tamaki. A satisfiability algorithm for depth two circuits with a sub-quadratic number of symmetric and threshold gates. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:100, 2016.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.

[TV07] Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007.

[TX13] Luca Trevisan and Tongke Xue. A Derandomized Switching Lemma and an Improved Derandomization of AC^0. In *Conference on Computational Complexity* (CCC), pages 242–247, 2013.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[Vio20] Emanuele Viola. New lower bounds for probabilistic degree and AC^0 with parity gates. 2020.

[Vol14] Ilya Volkovich. On learning, lower bounds and (un) keeping promises. In *International Colloquium on Automata, Languages, and Programming*, pages 1027–1038. Springer, 2014.

[VW20] Nikhil Vyas and R. Ryan Williams. Lower bounds against sparse symmetric functions of ACC circuits: Expanding the reach of $\#\mathsf{SAT}$ algorithms. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 59:1–59:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[Wil11] Ryan Williams. Guest column: a casual tour around a circuit complexity bound. *ACM SIGACT News*, 42(3):54–76, 2011.

[Wil13a] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013.

[Wil13b] Ryan Williams. Towards NEXP versus BPP? In *International Computer Science Symposium in Russia*, pages 174–182. Springer, 2013.

[Wil14a] Ryan Williams. Algorithms for circuits and circuits for algorithms. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 248–261. IEEE, 2014.

[Wil14b] Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 194–202. ACM, 2014.

[Wil14c] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):1–32, 2014.

[Wil16] R. Ryan Williams. Natural proofs versus derandomization. *SIAM J. Comput.*, 45(2):497–529, 2016.

[Wil18] Richard Ryan Williams. Limits on representing Boolean functions by linear combinations of simple functions: Thresholds, relus, and low-degree polynomials. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 6:1–6:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *Symposium on Foundations of Computer Science* (FOCS), pages 1–10, 1985.

[Yao89] Andrew Chi-Chih Yao. Circuits and local computation. In *Symposium on Theory of Computing* (STOC), pages 186–196, 1989.

[Yap83] Chee K Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical computer science*, 26(3):287–300, 1983.