Imperial College London
Department of Computing

The Complexity and Applications of Circuit Minimization

Dimitrios Myrisiotis

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computing.

Στην μητέρα μου, Καλλιόπη (Πόπη) Φλώρου,

και την Μνήμη του πατέρα μου, Ηλία Μυρισιώτη.

To my mother, Kalliopi (Popi) Florou,

and the Memory of my father, Ilias Myrisiotis.

Statement of Originality

I declare that the material presented in this thesis is my own work except where otherwise stated and referenced according to the established practices of the field.

Copyright Declaration

Acknowledgements

I want to thank *many* people for their support and help during my PhD studies. First, I want to thank my advisors and coauthors (presented in the order of meeting them).

- *Mahdi Cheraghchi.* Mahdi gave me everything I could have ever asked as a PhD student. Above all, he gave me intellectual freedom and unconditional trust. I consider myself very fortunate to have Mahdi as my PhD advisor.

- *Mario Berta.* Mario, as my second supervisor, was always there for me when I needed guidance or advice. Thank you, Mario!

- *Igor C. Oliveira.* Igor was the first to meet from the circuit complexity community. He immediately made me feel welcome! I want to thank Igor for our countless discussions and for our collaboration [63].

- *Valentine Kabanets.* Valentine is a very generous person and has great research ideas. I am very grateful for Valentine's help during my PhD studies, and our productive collaboration [29, 63]. I also want to thank him for offering me an internship at Simon Fraser University (SFU) during the Summer of 2018. I had a great time there, and learned a lot! I remember I used to explore the SFU campus in the evenings. If you ever go there, then you should visit the academic quadrangle!

- *Zhenjian Lu.* During my PhD studies, Zhenjian was also like an advisor to me. I am thankful to have met Zhenjian in Vancouver, in 2018. I have learned so much from him about complexity. I wish him the best in his very promising academic career!

- *Yuichi Yoshida.* Yuichi is a very smart and humble person. His work ethic is unmatched. I feel privileged to have worked with Yuichi [28]. I want to thank him for offering me an internship at National Institute of Informatics during the Spring and Summer of 2019: I had a great time there and explored Japan.

- *Shuichi Hirahara.* Shuichi is a very patient person. I want to thank Shuichi for putting up with me during our collaboration [28], and for correcting my so frequent mistakes. It was really special to work with Shuichi; he is one of a kind!

- *Eric Allender.* It was a pleasure to learn from Eric. I am very grateful for having the opportunity to work with him [7]! Eric has a great perception of complexity theory, and his point of view is always unique. Thank you very much, Eric!

- *Sajin Koroth.* I want to thank Sajin for his help during (and after!) our collaboration. In particular, I want to thank Sajin for being so patient with me while I was rehearsing my CCC 2020 talk.

- *Ilya Volkovich.* Ilya is a great person. I want to thank Ilya for working with me [7], and for offering tons of chocolate to Mahdi and me when we were discussing MCSP in his office. Unfortunately, I was (trying to be) sugar-free at the time, and never had any! Thanks, Ilya!

- *Harsha Tirumala.* I want to thank Harsha for our inspiring discussions and productive collaboration [7].

I also want to thank the following people.

- *João Ribeiro.* João is a good friend. We started our PhDs together in 2017, both having Mahdi as our advisor, and have shared to a great extent the PhD experience. I am very grateful to have met João, and I want to thank him for his precious help! João is a great person and researcher, and his work is very influential.

- *Ninad Rajgopal.* I want to thank my good friend Ninad for all our inspiring discussions throughout our PhD studies, and for his hospitality during the many times I visited Oxford.

- *Francesco Borderi.* Francesco is one of the most optimistic people that I have ever met. I am happy to know Francesco, and I wish him good luck in his endeavors in the finance industry!

- *Sam Werner.* I want to thank Sam for getting porridge with me in the mornings, at the café of Senior Common Room, and for helping me find my way around the gym! Thanks for the help, Sam! I also want to thank him for letting me know that, in the US Midwest, dinner is "supper" and soft drink is "pop."

- *Katerina Koutsouri.* I want to thank Katerina for all the nice coffee breaks that we shared outside ACEX, and for all the advice and help!

- *Dominik Harz.* Dominik was the second person to meet at Imperial College London after I arrived. We shared the same office and (almost) the same sense of humor. I will miss Dominik!

- *Alexei Zamyatin.* Alexei is one of the smartest people I know, and has a great entrepreneurial spirit. I want to thank Alexei for helping cultivate a very inspiring office

culture, and for explaining to me what is *the Blockchain*.

- *Daniel Perez.* Daniel is a computer wizard! Definitely the most computer literate person I know. I admire Daniel for making his hobby a job, in such a successful way. I want to thank Daniel for all the late-night dinners that we shared while at Imperial.

- *Charles You.* Charles is definitely a very strong personality, and has great a sense of humor. I will definitely miss chatting with Charles.

I would now like to thank the following people for their help in the development of the publications of which I was a coauthor.

I want to thank Emanuele Viola and Osamu Watanabe for bringing to our attention the works by Kalyanasundaram and Schnitger [64] and Watanabe [115], respectively, and for helpful discussions. In particular, I thank Emanuele Viola for explaining to us his works [43, 113]. I thank Chin Ho Lee for answering our questions regarding his work [67]. I thank Rahul Santhanam for pointing out that Nečiporuk's method can be applied to not only MKtP but also MKTP (Theorem 1.13), and I thank Paul Beame for bringing his work [17] to our attention.

Moreover, I would like to thank Russell Impagliazzo for explaining his work [56] to us, and Ján Pich for illuminating discussions. I also thank Ján Pich for bringing his work [90] to our attention. I thank Mikito Nanashima and Hanlin Ren for helpful comments and for spotting bugs in the proofs of an earlier version of Theorem 10.28, respectively. In particular, I thank Hanlin Ren for asking the question of whether KT complexity would be an appropriate complexity measure to consider in the context of Chapter 10. In this regard, I would also like to thank Yanyi Liu and Rafael Pass for the excellent correspondence regarding their work [70, 73].

I thank Iain Phillips for our nice collaboration in teaching *Complexity* and *Graphs and Algorithms* at Imperial College London. In the same vein, I thank Steffen van Bakel and Herbert

Wiklicky for our collaboration in teaching *Discrete Mathematics* and *Quantum Computing*, respectively.

I thank Iddo Tzameret, Rahul Santhanam, and Iain Phillips for checking this work for errors, and for their useful and constructive feedback.

I am very thankful to one of our ACM ToCT reviewers for improving our $2^{N/\widetilde{O}(\log^2 N)}$ MCSP size lower bound against DNFs [29] to an optimal $2^{\Omega(N)}$ (see Theorem 1.4 and Theorem 5.5), where N is the size of the input truth table to MCSP.

Some of the work included in this thesis was partly carried out during a visit to DIMACS, with support from the Special Focus on Lower Bounds in Computational Complexity funded under NSF Grant CCF-1836666.

I want to thank Jaeyoung (Jae) Choi for everything that we have shared: The distance, and the love. Finally, I want to thank my mother, Popi, for her unconditional support throughout my PhD studies, and her courageous and inspiring outlook on life.

List of Publications

The material presented in this thesis is based on (parts of) the following works.

1. Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. Manuscript, 2021.

   The collaboration that led to this manuscript started when I visited Prof. Eric Allender for an internship (Summer 2020) in Rutgers University (New Brunswick, NJ, USA).

2. Mahdi Cheraghchi, Shuichi Hirahara, Dimitrios Myrisiotis, and Yuichi Yoshida. One-tape Turing machine and branching program lower bounds for MCSP. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 23:1–23:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

   The collaboration that led to this publication started when I visited Prof. Yuichi Yoshida for an internship (Summer 2019) in the National Institute of Informatics (Tokyo, Japan).

3. 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 Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July*

28-31, 2020, Saarbrücken, Germany (Virtual Conference), volume 169 of *LIPIcs*, pages 15:1–15:41. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

I was invited to participate in the project that led to this publication after a significant amount of time has passed since it started. For that matter, there are only two results appearing here from this publication (see Chapter 7).

4. Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. *ACM Trans. Comput. Theory*, 12(3):21:1–21:27, 2020.

   The collaboration that led to this publication started when I visited Prof. Valentine Kabanets for an internship (Summer 2018) in Simon Fraser University (Burnaby, BC, Canada).

Abstract

The main topic of this thesis is the study of the Minimum Circuit Size Problem (MCSP). MCSP is a decision problem: Given the truth table of a finite Boolean function and a size parameter $0 \leq \theta \leq 2^n - 1$ (in binary), is there a Boolean circuit C of size at most θ such that C computes f? MCSP is an important problem due to its unexpected connections to many other areas of complexity theory, such as learning, pseudorandomness, and average-case complexity.

In a similar manner, Kolmogorov complexity poses the following question: Given a string x, what is the size of the smallest program that outputs x when run on the empty string? While Kolmogorov complexity cannot be computed, in general, there are resource-bounded versions of Kolmogorov complexity that are computable and have been very influential in complexity theory, just like MCSP is.

In this work, we study MCSP and some of its Kolmogorov complexity counterparts, and answer the following two questions.

1. *What is the complexity of* MCSP*? That is, how easy or difficult is it to compute* MCSP*?*
   We prove MCSP lower bounds against several restricted models of computation. These, among others, include formulas, branching programs, constant-depth circuits, and one-tape Turing machines. Almost all of the above lower bounds are the first of their kind for MCSP, and almost match the state-of-the-art lower bounds against these models.

2. *What are some interesting applications of* MCSP*?* Here, we show that the average-case hardness of a conditional Kolmogorov complexity counterpart of MCSP is *almost* equivalent to the existence of one-way functions (OWFs). Along with the concurrent and independent work by Liu and Pass [73], this is one of the first natural NP-complete problems whose average-case hardness is shown to be (almost) equivalent to the existence of OWFs.

List of Common Notation and Abbreviations

$g(n) = o(f(n))$	denotes the fact that $\lim_{n\to\infty} g(n)/f(n) = 0$
$g(n) = \omega(f(n))$	denotes the fact that $f(n) = o(g(n))$
$g(n) = O(f(n))$	denotes the fact that $\lim_{n\to\infty} g(n)/f(n)$ is bounded by some constant
$g(n) = \Omega(f(n))$	denotes the fact that $f(n) = O(g(n))$
$g(n) = \widetilde{O}(f(n))$	denotes the fact that $g(n) = O(f(n)\,\mathrm{polylog}(f(n)))$
$g(n) = \widetilde{\Omega}(f(n))$	denotes the fact that $g(n) = \Omega(f(n)/\mathrm{polylog}(f(n)))$
$\mathbb{N}$	denotes the set of natural numbers
$\mathbb{R}$	denotes the set of real numbers
$\mathbb{R}[x_1, \ldots, x_n]$	denotes the ring of polynomials with real coefficients over the variables $x_1, \ldots, x_n$
$\mathbb{R}_{>0}$	denotes the set of positive real numbers
$\mathbb{F}_n$	denotes the field of n elements, where n is a prime power
PRG	stands for pseudorandom generator
UTM	stands for universal Turing machine

DTM	stands for deterministic (one-tape) Turing machine
RTM	stands for randomized (one-tape) Turing machine
$A \times B$	denotes the Cartesian product of the sets A and B
$\lvert x \rvert$	if x is a string, then $\lvert x \rvert$ denotes the length of x; if x is a real number, then $\lvert x \rvert$ denotes the absolute value of x
$\mathbf{Pr}[E]$	denotes the probability that event E occurs
$\mathbf{E}[X]$	denotes the expectation of the random variable X
$x \overset{\varepsilon}{\approx} y$	for $x, y \in [0,1]$ and $0 \leq \varepsilon \leq 1$, this denotes the fact $\lvert x - y \rvert \leq \varepsilon$
$f\vert_\rho$	for $f : \{0,1\}^n \to \{0,1\}$ and $\rho \in \{0,1,*\}$, this denotes the restricted version of f according to the restriction ρ
$\mathtt{tt}(f)$	denotes the truth table $f(0^n)\, f\big(0^{n-1}1\big) \cdots f(1^n)$ of the Boolean function $f : \{0,1\}^n \to \{0,1\}$; in this case we may also say that $\mathtt{tt}(f)$ *encodes* f
$\mathrm{CC}(f)$	denotes the circuit complexity of the Boolean function f, that is, the size of the minimum-size Boolean circuit that computes f
MCSP	denotes the Minimum Circuit Size Problem
$\mathrm{MCSP}[\theta]$	for a function $\theta : \mathbb{N} \to \mathbb{N}$, this denotes the Minimum Circuit Size Problem parameterized by θ
λ	denotes the empty string, or a function associated with the circuit complexity of (the Boolean function encoded by) the output of a PRG; this will be clear from context
$L(f)$	depending on the context, this denotes the size of the minimum-size formula that computes f (leaf-size), or the size of the minimum-size branching program that computes f

AC^0	denotes the class of languages that can be computed by a collection of constant-depth polynomial-size Boolean circuits
$\mathsf{SIZE}[s]$	for a function $s : \mathbb{N} \to \mathbb{N}$, this denotes the class of languages that can be computed by a collection of size-$O(s(n))$ Boolean circuits
P/poly	denotes the class of languages that can be computed by a collection of polynomial-size Boolean circuits
$\mathsf{FORMULA}[s]$	for a function $s : \mathbb{N} \to \mathbb{N}$, this denotes the class of languages that can be computed by a collection of size-$O(s(n))$ De Morgan formulas
$\mathsf{FORMULA}[s] \circ \mathcal{G}$	for a function $s : \mathbb{N} \to \mathbb{N}$, this denotes the class of languages that can be computed by a collection of size-$O(s(n))$ De Morgan formulas of functions from the class $\mathcal{G}$
$\mathsf{FORMULA}[s] \circ \mathsf{XOR}$	for a function $s : \mathbb{N} \to \mathbb{N}$, this denotes the class of languages that can be computed by a collection of size-$O(s(n))$ De Morgan formulas of parities
P	the class of languages that can be computed in polynomial time by a deterministic Turing machine
NP	the class of languages that can be computed in polynomial time by a non-deterministic Turing machine
$\mathsf{DTIME}_1[t]$	for a function $t : \mathbb{N} \to \mathbb{N}$, this denotes the class of languages that can be computed in time $O(t(n))$ by a one-tape Turing machine, that has a one-way read-only input tape

$\mathsf{BPTIME}_1[t]$	for a function $t : \mathbb{N} \to \mathbb{N}$, this denotes the class of languages that can be computed in time $O(t(n))$ by a bounded-error probabilistic one-tape Turing machine, that has a one-way read-only input tape and a one-way read-only random tape
$\mathrm{K}(x)$	denotes the Kolmogorov complexity of a string x
$\mathrm{K}(x \mid y)$	denotes the Kolmogorov complexity of a string x given a string y
$\mathrm{KT}(x)$	denotes the KT complexity of a string x
$\mathrm{KT}(x \mid y)$	denotes the KT complexity of a string x given a string y
MKTP	denotes the Minimum KT-complexity Problem
MKTP$[\theta]$	for a function $\theta : \mathbb{N} \to \mathbb{N}$, this denotes the Minimum KT-complexity Problem parameterized by θ
McKTP	denotes the Minimum Conditional KT-complexity Problem

List of Figures

Table of Contents

Chapter 1

Introduction

Given the truth table of some Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a size parameter $0 \leq \theta \leq 2^n - 1$ (in binary), the Minimum Circuit Size Problem (MCSP) asks whether f can be computed by a Boolean circuit of size at most θ. Understanding the exact complexity of MCSP is an important open problem in computational complexity theory, dating back to the 1950s [108].

MCSP is a fundamental problem in complexity theory because of its surprising connections to many research areas, such as circuit complexity [93, 62, 50, 79, 49, 10], learning theory [23], and cryptography [93, 46, 47]. It is straightforward to see that MCSP $\in \mathsf{NP}$ because, given a circuit C of size s as a certificate, one can check whether C computes the given function f in time $N^{O(1)}$. On the other hand, the NP-completeness of MCSP is a long-standing open question, which dates back to the introduction of the theory of NP-completeness (cf. [12]), and it has an application to the equivalence between the worst-case and average-case complexity of NP (cf. [47]).

A popular conjecture is that MCSP is NP-hard. However, despite serious efforts over the

years, such a proof is still unknown. Given that it is difficult to show that MCSP is hard, perhaps the problem is easy? It turns out that this cannot be the case under some plausible cryptographic assumptions. More specifically, it is known that if one-way functions exist, then MCSP is not in P [62]. As proving an *unconditional* lower bound for MCSP seems far beyond the reach of currently known techniques, can we at least prove unconditional lower bounds for MCSP against some *restricted* computational models?

To this end, Razborov and Rudich [93] showed that there exists no AC^0-natural property useful against $\mathsf{AC}^0[\oplus]$, which in particular implies that MCSP $\notin \mathsf{AC}^0$. Hirahara and Santhanam [48] proved that MCSP essentially requires quadratic-size de Morgan formulas, and Golovnev, Ilango, Impagliazzo, Kabanets, Kolokolova, and Tal [40] proved that, for any prime p, MCSP requires constant-depth circuits augmented with MOD_p gates of weakly-exponential size.

Expanding the line of work outlined above (that is, to prove unconditional lower bounds for MCSP against restricted models of computation) is the topic of the first part of this thesis; see Section 1.1 and Section 1.2.

In what follows, for any function $s : \mathbb{N} \to \mathbb{N}$, we denote by MCSP$[s(n)]$ the parameterized version of MCSP, whereby the size parameter θ is not part of the input but fixed to $s(n)$.

1.1 The Complexity of MCSP

1.1.1 MCSP Lower Bounds Against Formulas and Branching Programs

Among the most studied restricted computational models in complexity theory are the models of De Morgan formulas and branching programs. For De Morgan formulas, the state-of-the-art lower bound is almost cubic, namely $N^{3-o(1)}$, for some polynomial-time computable

function [45, 104, 106, 33]. For branching programs, the state-of-the-art lower bound is almost quadratic, namely $N^{2-o(1)}$, for some polynomial-time computable function [82].

For De Morgan formulas, as mentioned earlier, Hirahara and Santhanam [48] showed that computing MCSP requires De Morgan formulas of size $N^{2-o(1)}$. Prior to this work, there were no MCSP lower bounds against branching programs.

It is natural then to ask whether we can get MCSP lower bounds against De Morgan formulas and branching programs that match the state-of-the-art lower bounds against these models. More specifically, can we show that computing MCSP requires De Morgan formulas of size $N^{3-o(1)}$ and branching programs of size $N^{2-o(1)}$?

Our first result is an almost-cubic De Morgan formula lower bound for MCSP.

Theorem 1.1. *Any De Morgan formula computing* $\mathrm{MCSP}\left[2^{\Omega(n)}\right]$ *on truth tables of length* $N := 2^n$ *must have size at least* $N^3/2^{O\left(\log^{2/3} N\right)}$.

We also get almost-quadratic lower bounds against formulas over an arbitrary basis as well as general branching programs; these almost match the best-known lower bounds against these models [82].

Theorem 1.2. *Let* C *be either a formula over any (bounded fan-in) basis or a branching program that computes* $\mathrm{MCSP}\left[2^{\Omega(n)}\right]$ *on truth tables of length* $N := 2^n$. *Then* C *must have size at least* $N^2/2^{O\left(\sqrt{\log N}\right)}$.

1.1.2 MCSP Lower Bounds Against Constant Depth Circuits

Constant-depth circuits are a popular model of computation that has been studied extensively since the 80s. The strongest known lower bound against constant-depth circuits (i.e., the class AC^0) is for the Parity function, which on input $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ takes value 1 if and

only if $\sum_{i=1}^{n} x_i \bmod 2 = 1$. In particular, Parity on N variables requires depth-d AC^0 circuits of size $2^{\Omega\left(N^{1/(d-1)}\right)}$ [44]. Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [6] showed that MCSP, on functions represented as truth tables of length N, cannot be computed by polynomial-size constant-depth AC^0 circuits. In fact, by a more careful analysis of their argument, one can get a lower bound of $2^{N^{1/(c\cdot d+O(1))}}$, for a constant $c \geq 2$. However, such a lower bound still has a worse dependence on the depth compared to the Parity lower bound. So it is natural to ask: Can one show that computing MCSP requires depth-d AC^0 circuits of size $2^{N^{1/(d+O(1))}}$?

We present the following improved lower bound for MCSP, whose dependence on the depth matches the one in the Parity lower bound, up to a small additive constant.

Theorem 1.3. *For every $d > 2$ and every constant $\gamma > 0$, any depth-d AC^0 circuit computing* MCSP *on truth tables of length N must have size $2^{\Omega\left(N^{1/(d+1+\gamma)}\right)}$.*

For the special case of depth-2 circuits, we present an optimal lower bound.

Theorem 1.4. *Any* CNF *or* DNF *computing* MCSP *on truth tables of length N must have size $2^{\Omega(N)}$.*

1.1.3 MCSP Lower Bounds Against Augmented Constant Depth Circuits

Another restricted type of circuits that are well-studied in circuit complexity is the class of constant-depth circuits augmented with few SYM (symmetric) or THR (linear threshold) gates (see, e.g., [76, 112, 75, 100]). A SYM gate computes a symmetric function, which is a Boolean function whose output depends only on the sum of its input variables. A THR gate computes a linear threshold function, which is a Boolean function defined as the sign of some linear form, on Boolean variables, with real coefficients.

We get MCSP lower bounds against such circuits, by giving a fine-grained analysis of the approach of obtaining MCSP lower bounds from average-case hardness via the Nisan-Wigderson framework [83] (see Section 6.1). Our results match the best-known lower bounds against these circuits, due to Servedio and Tan [100].

Theorem 1.5. *There exists a constant $\gamma > 0$ such that the following holds. Let $\mathfrak{C}$ be the class of constant-depth N-input AC^0 circuits augmented with at most $2^{\gamma\sqrt{\log N}}$ SYM or THR gates. Then any circuit in $\mathfrak{C}$ computing $\mathrm{MCSP}\left[2^{\Omega(n)}\right]$ on truth tables of length $N := 2^n$ must have size $N^{\Omega\left(\sqrt{\log N}\right)}$.*

Remark 1.6. Note that the class $\mathfrak{C}$ of Theorem 1.5 is very expressive, subsuming $\mathsf{THR} \circ \mathsf{MAJ}$ circuits of small size, and other interesting classes.

1.1.4 MCSP Lower Bounds Against De Morgan Formulas of Parities

As briefly mentioned above, in Section 1.1.1, Boolean formulas are a well-studied model of computation. Indeed, Boolean formulas have been investigated since the early years of complexity theory (see, e.g., [103, 82, 65, 15, 89, 59, 45, 104, 33]). The techniques underlying these complexity-theoretic results have also enabled algorithmic developments. These include learning algorithms [95, 99], satisfiability algorithms (cf. [105]), compression algorithms [27], and the construction of pseudorandom generators (PRGs) [57] for Boolean formulas of different sizes. Nonetheless, despite many decades of research, the current non-trivial algorithms and lower bounds apply only to formulas of less than cubic size, and understanding larger formulas remains a major open problem in circuit complexity.

In many scenarios, however, understanding smaller formulas whose leaves are replaced by certain functions would also be very useful. Some examples are the following. In what follows,

we let $\mathsf{FORMULA}[s] \circ \mathcal{G}$ denote the set of Boolean functions computed by formulas containing at most s leaves, where each leaf computes a function in $\mathcal{G}$.

- Oliveira, Pich, and Santhanam [86] show that proving certain lower bounds against formulas of size $n^{1+\varepsilon}$ over parity (XOR) gates would have significant consequences in complexity theory. Note that De Morgan formulas of size $n^{3+\varepsilon}$ can simulate such devices. Therefore, a better understanding of the $\mathsf{FORMULA} \circ \mathsf{XOR}$ model is *necessary* before we are able to analyze super-cubic size formulas.[1]

- Tal [106] obtains almost quadratic lower bounds for the model of bipartite formulas, where there is a fixed partition of the input variables into $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$, and a formula leaf can compute an *arbitrary* function over either $\vec{x}$ or $\vec{y}$. This model was originally investigated by Pudlák, Rödl, and Savický [92], where it was referred to as graph complexity. The model is also equivalent to PSPACE-protocols in communication complexity (cf. [41]).

- Abboud and Bringmann [1] consider formulas where the leaves are threshold gates whose input wires can be arbitrary functions applied to either the first or the second half of the input. This extension of bipartite formulas is denoted by $\mathcal{F}_2$ in [1]. Their work establishes connections between faster $\mathcal{F}_2$-SAT algorithms, the complexity of problems in P such as Longest Common Subsequence and the Fréchet Distance Problem, and circuit lower bounds.

- Polytopes (i.e., intersections of half-spaces), which correspond to $\mathcal{G}$ being the class of linear-threshold functions, and the formula contains only AND gates as internal gates.

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

The construction of PRGs for this model has received significant attention in the literature (see [85] and references therein).

In this work, we give a novel pseudorandom generator (PRGs) that fools $\mathsf{FORMULA} \circ \mathsf{XOR}$.

Recall that a PRG that fools a class of functions $\mathfrak{C}$ is a function G mapping short strings (seeds) to longer strings so that every function in $\mathfrak{C}$ accepts the output of G on a uniformly random seed with about the same probability as that for an actual uniformly random string. More formally, $G : \{0,1\}^{\ell} \to \{0,1\}^{n}$ is a PRG that ε-fools $\mathfrak{C}$ if for every Boolean function $h : \{0,1\}^{n} \to \{0,1\}$ in $\mathfrak{C}$ we have

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

The parameter $\ell = \ell(n)$ is called the seed length of the PRG and is the main quantity to be minimized when constructing PRGs.

There exists a PRG that fools formulas of size s and has a seed of length $s^{1/3+o(1)}$ [57]. In particular, there are non-trivial PRGs for n-variate formulas of size nearly n^3. Unfortunately, such PRGs cannot be used to fool even linear size formulas over parity functions, since the naive simulation of these augmented formulas by standard Boolean formulas requires size n^3. Moreover, it is not hard to see that this simulation is optimal: Andreev's function, which is hard against formulas of nearly cubic size (cf. [45]), can be easily computed in $\mathsf{FORMULA}[O(n)] \circ \mathsf{XOR}$. Given that a crucial idea in the construction of the PRG in [57] (shrinkage under restrictions) comes from this lower bound proof, new techniques are needed in order to approach the problem in the $\mathsf{FORMULA} \circ \mathsf{XOR}$ model.

More generally, extending a computational model for which strong PRGs are known to allow parities at the bottom layer can cause significant difficulties. A well-known example is AC^0 circuits and their extension to $\mathsf{AC}^0 \circ \mathsf{XOR}$. While the former class admits PRGs of

poly-logarithmic seed length (see, e.g., [101]), the most efficient PRG construction for the latter model has seed length $(1 - o(1)) \cdot n$ [36]. Consequently, designing PRGs of seed length at most $(1 - \Omega(1)) \cdot n$ can already be a challenge. We are not aware of previous results on PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$ for any non-trivial class $\mathcal{G}$.

We construct the following novel PRG that fools $\mathsf{FORMULA} \circ \mathsf{XOR}$ that has almost quadratic stretch.

Theorem 1.7. *There is a PRG that ε-fools n-input formulas of parities from* $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ *and has seed length*

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

Under a standard connection between PRGs and lower bounds (see, e.g., [61]), improving the dependence on s in the seed length for $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ would require the proof of super-quadratic lower bounds against $\mathsf{FORMULA} \circ \mathsf{XOR}$.

Prior to this work, the only known lower bound against $\mathsf{FORMULA} \circ \mathsf{XOR}$ (or bipartite formulas) was the recent result of Tal [106] showing that the inner-product function IP_n is hard (even on average) against formulas of almost quadratic size.

We present the following new almost quadratic-size lower bound against $\mathsf{FORMULA} \circ \mathsf{XOR}$ by combining Theorem 1.7 with the framework of Chapter 3 (which shows a way of obtaining MCSP lower bounds against a computational model $\mathcal{M}$, by making use of PRGs that fool $\mathcal{M}$).

Theorem 1.8. *If* $\mathrm{MCSP}\big[2^{\Omega(n)}\big]$ *on truth tables of length* $N := 2^n$ *is in* $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$, *then* $s = \widetilde{\Omega}(N^2)$.

MCSP plays an important role in the theory of hardness magnification (see [86, 26] and earlier discussion). In particular, if one could show that $\mathrm{MCSP}\big[2^{o(n)}\big]$ is not in $\mathsf{FORMULA}[n^{1+\varepsilon}] \circ \mathsf{XOR}$ for some $\varepsilon > 0$, then it would follow that NP cannot be computed by Boolean formulas

of size n^c, where $c \in \mathbb{N}$ is arbitrary. Theorem 1.8 makes partial progress in this direction by establishing the first lower bound for this problem in the $\mathsf{FORMULA} \circ \mathsf{XOR}$ model.

1.1.5 MCSP Lower Bounds Against One-tape Turing Machines

As noted in Section 1.1.4, hardness magnification phenomena have exhibited surprising connections between very weak lower bounds of MCSP and important open questions of complexity theory. Oliveira and Santhanam [88] (later with Pich [86]) showed that, if an approximation version of MCSP cannot be computed by a circuit of size $N^{1.01}$, then $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ (in particular, $\mathsf{P} \neq \mathsf{NP}$ follows). Similarly, McKay, Murray, and Williams [77] showed that, if MCSP$[s(n)]$ cannot be computed by a one-pass streaming algorithm of $\mathrm{poly}(s(n))$ space and $\mathrm{poly}(s(n))$ update time, then $\mathsf{P} \neq \mathsf{NP}$. Therefore, in order to obtain a breakthrough result, it is sufficient to obtain a very weak lower bound for MCSP.

Are hardness magnification phenomena plausible approaches for resolving the P versus NP question? We do not know the answer yet. However, it should be noted that, as argued in [11, 88], hardness magnification phenomena appear to bypass the *natural proof* barrier of Razborov and Rudich [93], which is one of the major barriers of complexity theory for resolving the P versus NP question. Most lower bound proof techniques of complexity theory are "natural" in the following sense: Given a lower bound proof for a circuit class $\mathfrak{C}$, one can interpret it as an efficient average-case algorithm for solving $\mathfrak{C}$-MCSP (i.e., one can efficiently decide whether a given Boolean function f can be computed by a small $\mathfrak{C}$-circuit when the input f is chosen uniformly at random; cf. Hirahara and Santhanam [48]). Razborov and Rudich [93] showed that such a "natural proof" technique is unlikely to resolve $\mathsf{NP} \not\subseteq \mathsf{P/poly}$; thus we need to develop fundamentally new proof techniques. There seems to be no simple argument that naturalizes proof techniques of hardness magnification phenomena; hence, investigating

hardness magnification phenomena could lead us to a new non-natural proof technique.

Motivated by hardness magnification phenomena, we study the time required to compute MCSP by using a one-tape Turing machine. We first observe that the hardness magnification phenomena of [77] imply that a barely superlinear time lower bound for a one-tape Turing machine is sufficient for resolving the P versus NP question.

Theorem 1.9 (A corollary of McKay, Murray, and Williams [77]; see Section 8.2)**.** *There exists a small constant $\mu > 0$ such that if $\mathrm{MCSP}[2^{\mu \cdot n}] \notin \mathsf{DTIME}_1\left[N^{1.01}\right]$, then $\mathsf{P} \neq \mathsf{NP}$.*

Here, we denote by $\mathsf{DTIME}_1[t(N)]$ the class of languages that can be computed by a Turing machine equipped with a one-way read-only input tape and a two-way read/write work tape running in time $O(t(N))$ on inputs of length N.

We note that it is rather counter-intuitive that there is a *universal* constant $\mu > 0$ satisfying the conditions of Theorem 1.9. For that matter, it is instructive to state Theorem 1.9 in the following logically equivalent way:

> If $\mathrm{MCSP}[2^{\mu \cdot n}] \notin \mathsf{DTIME}_1\left[N^{1.01}\right]$ for *all constants* $\mu > 0$, then $\mathsf{P} \neq \mathsf{NP}$.[2]

One of our results is a nearly quadratic lower bound on the time complexity of a *randomized* one-tape Turing machine (with one additional read-only one-way input tape) computing MCSP, as follows.

Theorem 1.10. *There exists some constant $0 < \mu < 1$ such that $\mathrm{MCSP}[2^{\mu \cdot n}]$ is not in $\mathsf{BPTIME}_1\left[N^{1.99}\right]$.*

Here, $\mathsf{BPTIME}_1[t(N)]$ denotes the class of languages that can be computed by a *two-sided-error randomized* Turing machine equipped with a one-way read-only input tape, a one-way

[2] Observe that $\exists \mu, (P(\mu) \Rightarrow Q)$ is logically equivalent to $\exists \mu, (\neg P(\mu) \vee Q)$, which is equivalent to $\neg(\forall \mu, P(\mu)) \vee Q$.

read-only random tape, and a two-way read/write work tape running in time $t(N)$ on inputs of length N; we say that a two-sided-error randomized algorithm *computes* a problem if it outputs a correct answer with high probability (say, with probability at least $2/3$) over the internal randomness of the algorithm.

Previously, no non-trivial lower bound on the time complexity required for computing MCSP by a Turing machine was known. Moreover, Theorem 1.10 essentially matches the best-known lower bound for this computational model; namely, the lower bound due to Kalyanasundaram and Schnitger [64], who showed that Element Distinctness is not in $\mathsf{BPTIME}_1\big[o\big(N^2/\log N\big)\big]$.

Our lower bound against $\mathsf{BPTIME}_1\big[N^{1.99}\big]$ is much stronger than the required lower bound (that is against $\mathsf{DTIME}_1\big[N^{1.01}\big]$) of the hardness magnification phenomenon of Theorem 1.9. However, Theorem 1.10 falls short of the hypothesis of the hardness magnification phenomenon of Theorem 1.9 because of the choice of the size parameter. In the hardness magnification phenomenon, we need to choose the size parameter to be $2^{\mu \cdot n}$ for some small constant $\mu > 0$, whereas, in our lower bound, we will choose μ to be some constant close to 1. That is, what is missing for proving $\mathsf{P} \neq \mathsf{NP}$ is to decrease the size parameter from $2^{(1-o(1))\cdot n}$ to $2^{o(n)}$ in Theorem 1.10, or to increase the size parameter from $2^{o(n)}$ to $2^{(1-o(1))\cdot n}$ in Theorem 1.9.

Next, we investigate the question of whether hardness magnification phenomena on MCSP$[s(n)]$ such as Theorem 1.9 can be proved when the size parameter $s(n)$ is large, as posed by Chen, Jin, and Williams [26]. As observed in [25], most existing proof techniques on hardness magnification phenomena are shown by constructing an oracle algorithm that makes short queries to some oracle. For example, behind the hardness magnification phenomena of Theorem 1.9 is a nearly-linear-time oracle algorithm that solves MCSP$[2^{o(n)}]$ by making queries of length $2^{o(n)}$ to some PH oracle (see Corollary 8.7 for a formal statement). Chen, Hirahara, Oliveira, Pich, Rajgopal, and Santhanam [25] showed that most lower bound proof techniques

can be generalized to such an oracle algorithm, thereby explaining the difficulty of combining hardness magnification phenomena with lower bound proof techniques. Following [25], we observe that our lower bound (Theorem 1.11) can be generalized to a lower bound against an oracle algorithm that makes short queries.

Theorem 1.11. *Let $O \subseteq \{0,1\}^*$ be any oracle. Then, for every constant $1/2 < \mu < 1$, $\mathrm{MCSP}[2^{\mu \cdot n}]$ on truth tables of size $N := 2^n$ is not in $\mathsf{BPTIME}_1^O\left[N^{1+\mu'}\right]$ for some constant $\mu' > 0$, where all of the strings queried to O are of length $N^{o(1)}$.*

Theorem 1.11 can be seen as a partial answer to a question posed by [26]: It is impossible to extend the hardness magnification phenomena of Theorem 1.9 to $\mathrm{MCSP}[2^{\mu n}]$ for $\mu > 1/2$ by using similar techniques used in [77]. Recall that the proof techniques behind [77] are to construct a nearly-linear-time oracle algorithm that solves $\mathrm{MCSP}[2^{\mu n}]$ by making short queries to some oracle; the existence of such an oracle algorithm is ruled out by Theorem 1.11 when $\mu > 1/2$. Therefore, in order to obtain a hardness magnification phenomenon for $\mathrm{MCSP}[2^{0.51n}]$, one needs to develop a completely different proof technique that does not rely on constructing an oracle algorithm that makes short queries.

1.2 The Complexity of MKTP

Kolmogorov complexity poses the following question: Given a string x, what is the size of the smallest program that outputs x when run on the empty string? Although this problem cannot be computed [69], in general, there are resource-bounded versions of it that are computable and also important for complexity theory. One of these variants is KT complexity.

For a string x, the KT complexity $\mathrm{KT}(x)$ is defined as the minimum value of $|M| + t$, over all programs M and integers t, such that for every $1 \leq i \leq |x|$ the machine M outputs the i-th

bit of x in time t, given the index i as input [6].

Remark 1.12. In the definition of KT complexity, above, the Turing machine M in question is required to output only one bit of x in time t, not all of x. For the sake of discussion, let us now define a related KT complexity measure, namely KT', such that $\mathrm{KT}'(x)$ is defined as the minimum value of $|M| + t$, over all programs M and integers t, such that on input the empty string the machine M outputs x in time t. How does KT complexity relate to KT' complexity? Let $s \in \mathbb{N}$ and observe the following.

1. If $\mathrm{KT}(x) \leq s$, then $\mathrm{KT}'(x) \leq |x| \cdot s$. This is so, because we can repeatedly use a machine M that generates any bit of x in time t to produce all of x in time $|x| \cdot t \leq |x| \cdot s$ (whereby $|M| + t \leq s$).

2. If $\mathrm{KT}'(x) \leq s$, then $\mathrm{KT}(x) \leq s + O(|x|)$. This is so, because given a machine that generates all of x, we can produce any bit of x just by reading x in time $O(|x|)$.

The two items above show that KT and KT' complexity are closely related. In what follows, we focus on the standard notion of KT complexity [6].

The Minimum KT-complexity Problem (MKTP) is the problem of deciding whether $\mathrm{KT}(x) \leq s$ given (x, s) as input. The complexity of MKTP is closely related to the complexity of MCSP [6], in that KT complexity is polynomially-related to circuit complexity. Moreover, there are many results in the literature that highlight the importance of MKTP. For example, given an oracle to MKTP, one can compute Graph Isomorphism in zero-error probabilistic polynomial time [9].

1.2.1 MKTP Lower Bounds Against Branching Programs

We prove the first MKTP lower bounds against branching programs by making use of *Nečiporuk's method*, which is a standard proof technique for proving lower bounds against branching programs. It appeared previously that Nečiporuk's method is not directly applicable to the problems such as MCSP [48]. In this work, we develop a new proof technique for applying Nečiporuk's method to MKTP.

Theorem 1.13. *The size of a branching program computing* MKTP *is at least* $\Omega(N^2/\log^2 N)$. *The size of a non-deterministic branching program or a parity branching program computing* MKTP *is at least* $\Omega(N^{1.5}/\log N)$.

Theorem 1.13 gives the first non-trivial lower bounds against non-deterministic and parity branching programs for MKTP and, in addition, these are the best lower bounds which can be obtained by using Nečiporuk's method (cf. [17]). Previously, by using a pseudorandom generator for branching programs constructed by [57], it was shown in [86] and Theorem 1.2 above that (deterministic) branching programs require $N^{2-o(1)}$ size to compute MCSP and MKTP.[3] Surprisingly, Theorem 1.13 is proved without using a pseudorandom generator.

1.3 Connections to Cryptography

In the second part of this thesis, we study some applications of a *conditional* variant of MKTP. This conditional variant of MKTP, termed McKTP, takes as input a string x, a size parameter s, and an auxiliary string y, and asks to decide whether $\mathrm{KT}(x \mid y) \leq s$ or not. (The term

[3] It is worthy of note that Theorem 1.13 mildly improves the lower bounds of [86] and Theorem 1.2 to $\Omega(N^2/\log^2 N)$ by directly applying Nečiporuk's method, which matches the state-of-the-art lower bound for any explicit function up to a constant factor.

"conditional" comes from the presence of the auxiliary string y in the definition of McKTP, as we are "conditioning on knowing y.") The applications of McKTP pertain to cryptography, and in particular are about one-way functions (OWFs).

OWFs —that is, functions that are easy to compute but hard to invert— are objects of great importance in cryptography and computational complexity. For example, it is known that OWFs exist if and only if pseudorandom generators exist [46] and, moreover, if OWFs exist, then $\mathsf{P} \neq \mathsf{NP}$.

We ask the following question:

> *Can the existence of OWFs be shown to be equivalent to the average-case hardness of some natural* NP*-complete problem over the uniform distribution?*

We take concrete steps toward giving an affirmative answer to this question, by presenting a candidate problem.

Remark 1.14. In what follows, "hard-on-average" will mean "hard-on-average over the uniform distribution;" unless the underlying distribution is otherwise specified.

Note that by Impagliazzo and Naor [58] it is known that there exists some NP-complete problem (Subset Sum) whose average-case hardness implies the existence of OWFs. However, what we attempt to do is different: We want to make concrete progress in *characterizing* OWFs by the average-case hardness of an NP-complete problem.

The importance of NP stems mainly from the fact that, for thousands of important naturally-occurring computational problems, their worst-case computational complexity is best explained by knowing that they are NP-complete. Liu and Pass [70] gave what is arguably the first "natural" example of a problem in NP that is hard-on-average if and only if OWFs exist, but this problem is not known to be NP-complete. The problem that Liu and Pass [70] considered

is a decision variant of computing time-bounded Kolmogorov complexity, namely K^t. Although it is not hard to modify their language to obtain an artificial NP-complete problem with the same average-case complexity (see Proposition A.1), there had been no "natural" example of an NP-complete problem whose average-case complexity is *equivalent* to the existence of OWFs. As mentioned above, our main contribution here is to make progress on this question.

Initially, we prove that the average-case hardness of McKTP implies the existence of OWFs.

Theorem 1.15 (Informal)**.** *OWFs exist if* McKTP *is hard-on-average on a polynomial fraction of its instances, over the uniform distribution.*

Theorem 1.15 suggests an approach for excluding Impagliazzo's *Pessiland* [55], that is, a version of our world where there are average-case hard problems in NP *and* there are no OWFs. This approach is based on the following observation. If McKTP is NP-hard under average-case reductions, then by Theorem 1.15 the existence of an average-case hard problem in NP would imply the existence of OWFs. Therefore proving that McKTP is NP-hard under average-case reductions excludes Pessiland.

Nevertheless, we prove that McKTP is NP-complete under randomized reductions.

Theorem 1.16 (Informal)**.** McKTP *is* NP*-complete under polynomial-time randomized one-side-error reductions.*

Finally, we prove a *weak* converse of Theorem 1.15.

Theorem 1.17 (Informal)**.** *OWFs exist only if* McKTP *is hard-on-average on an* exponential *fraction of its instances, over the uniform distribution.*

There has been a flurry of recent activity on this topic, and it may be helpful to present here the following timeline.

1. [70] is posted by Liu and Pass, relating the existence of OWFs to the average-case hardness of computing K^t complexity, as mentioned earlier.

2. [8] is posted by Allender, Cheraghchi, Myrisiotis, Tirumala, and Volkovich, claiming to characterize the existence of OWFs by the average-case complexity of an NP-complete problem called Sparse Partial MCSP (which is a polynomially-sparse variant of Partial MCSP). This paper was retracted.

3. [7] is posted by Allender, Cheraghchi, Myrisiotis, Tirumala, and Volkovich, presenting the proofs of Theorem 1.15 through Theorem 1.17 above.

4. [71] is posted by Liu and Pass, whereby they prove that *subexponentially-hard* OWFs exist if and only if MK^tP (Minimum K^t-complexity Problem; a decision problem based on K^t complexity) is average-case hard for *sublinear-time non-uniform* heuristics.

5. [74] is posted by Liu and Pass, showing that OWFs exist if and only if the EXP-complete language MKtP (Minimum Kt-complexity Problem; a decision problem based on Levin's notion of Kt complexity [68]) is hard-on-average, and that logspace-computable OWFs exist if and only if the PSPACE-complete language MKSP (Minimum KS complexity Problem; a decision problem based on KS complexity, which a space complexity analogue of KT complexity) is hard-on-average.

6. [97] is posted by Ren and Santhanam, showing that MKTP is hard-on-average if and only if logspace-computable OWFs exist.

7. [73] is posted (inspired by and in part as a response to [8]) by Liu and Pass, showing that a conditional variant of time-bounded Kolmogorov complexity is NP-complete, and is hard-on-average if and only if OWFs exist. The main points of difference between the

work by Liu and Pass [73] and that of Allender, Cheraghchi, Myrisiotis, Tirumala, and Volkovich [7], are that Liu and Pass consider conditional K^t complexity with respect to RAM programs, while Allender, Cheraghchi, Myrisiotis, Tirumala, and Volkovich consider conditional KT complexity with respect to oracle Turing machines, and that the work by Liu and Pass establishes an *equivalence* between the average-case hardness of an NP-complete problem and the existence of OWFs.

8. [54] is posted by Ilango, Ren, and Santhanam, showing that OWFs exist if and only if the undecidable problem MKP (Minimum Kolmogorov-complexity Problem; a decision variant of Kolmogorov complexity) is hard-on-average under a samplable distribution, and if and only if MCSP is hard-on-average under a locally-sampleable distribution.

9. [72] is posted by Liu and Pass, generalizing some of the results of Ilango, Ren, and Santhanam [54], whereby they show that there exists some sparse language L such that OWFs exist if and only if L is average-case hard with respect to some efficiently sampleable "high-entropy" distribution.

We now mention some other related (prior) work. An early goal in cryptographic research was to base the existence of cryptographically secure OWFs on the worst-case complexity of some NP-complete problem. This goal remains elusive; it was shown in [4] that no black-box argument of this sort can proceed based on non-adaptive reductions. Non-adaptive worst-case-to-average-case reductions were also studied by Bogdanov and Trevisan [19], who showed that such reductions to sets in NP exist only for problems in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. (This shows a weakness of this kind of reductions.) Recent work by Nanashima [80] holds open the possibility that the security of OWFs can be based on an *adaptive* black-box reduction, by first establishing a non-adaptive black-box reduction basing the existence of *auxiliary input one-way*

functions on the worst-case complexity of an NP-complete problem, although this would also require non-relativizing techniques. Instead of worst-case hardness, the focus of our work is on average-case hardness assumptions. A nice survey on this area, that lays out many of the issues about OWFs and average-case complexity, is the one by Bogdanov and Trevisan [18].

Hirahara and Santhanam have discussed zero-error average-case complexity of problems related to MKTP [48]. Santhanam [98] showed that a hitting set generator that outputs truth tables of low circuit complexity exists if and only if MCSP is zero-error average-case hard if and only if there exist pseudorandom distributions supported on truth tables of low circuit complexity. Hirahara also proved similar results connecting the worst-case and the zero-error average-case complexity of problems related to MCSP and Kolmogorov complexity [47].

More recently, Brzuska and Couteau [21] discuss basing OWFs on average-case hardness, stating that it remains an open question to do this for the general notion of average-case hardness. They present some negative results, indicating the difficulty of establishing the existence of fine-grained OWFs, based on the existence of average-case hardness, via black-box reductions.

There is also an important line of work (including Ajtai [2] and Micciancio and Regev [78]) basing the existence of OWFs on the *worst-case* complexity of certain problems in NP (including problems that are closely related to NP-complete problems, although they are not themselves known to be NP-complete).

1.4 Thesis Organization

This thesis is organized as follows.

- In Chapter 2 we present some background material.

- In Chapter 3 we present the main framework of obtaining lower bounds for MCSP.
- In Chapter 4 we prove Theorem 1.1 and Theorem 1.2.
- Chapter 5 contains the proofs of Theorem 1.3 and Theorem 1.4.
- Chapter 6 contains the proof of Theorem 1.5.
- Chapter 7 contains the proofs of Theorem 1.7 and Theorem 1.8.
- Chapter 8 contains the proofs of Theorem 1.9 and Theorem 1.11 (and for that matter the proof of Theorem 1.10 as well).
- Chapter 9 contains the proof of Theorem 1.13.
- Chapter 10 contains the proofs of Theorem 1.15, Theorem 1.16, and Theorem 1.17.
- In Chapter 11 we present some open problems.
- Finally, Appendix A contains the proof of Proposition A.1.

We conclude each chapter with some relevant notes.

Chapter 2

Preliminaries

We present in this chapter some background material. The reader is advised to have a look at the notation table at the beginning of this thesis (List of Common Notation and Abbreviations; page 19).

2.1 Boolean Functions and Computational Models

We identify infinite Boolean functions $f : \{0,1\}^* \to \{0,1\}$ with collections $\{f_n\}_{n\in\mathbb{N}}$, whereby $f_n : \{0,1\}^n \to \{0,1\}$ for all $n \in \mathbb{N}$.

We are interested in computing Boolean functions by making use of *computational models* that are realized by *devices*. For example, a Boolean circuit is a device (from the computational model of Boolean circuits) that computes a finite Boolean function.

The following are some of the computational models that we will encounter in this thesis.

2.1.1 Boolean Circuits

We consider Boolean circuits that are directed acyclic graphs (DAGs) over the *bounded* fan-in $\{\text{AND}, \text{OR}, \text{NOT}\}$ basis, where AND and OR have fan-in 2 and NOT has fan-in 1. The DAGs that are used in defining circuits have many sources (these can be variables, say $x_1, \ldots, x_n$, or constants such as 0 or 1) and one sink (this is the output). Given a circuit, its *size* is the number of its vertices (which are called gates).

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the *circuit complexity of* f, denoted $\text{CC}(f)$, is the size of a minimum size circuit that computes f. For a size function $s : \mathbb{N} \to \mathbb{N}$, we denote by $\mathsf{SIZE}[s(n)]$ the class of Boolean functions $f = \{f_n\}_{n\in\mathbb{N}}$, whereby $f_n : \{0,1\}^n \to \{0,1\}$ for all $n \in \mathbb{N}$, such that $\text{CC}(f_n) \leq s(n)$ for all $n \in \mathbb{N}$. For a string $y \in \{0,1\}^{2^n}$, we denote by $\text{CC}(y)$ the circuit complexity of the function $f_y : \{0,1\}^n \to \{0,1\}$ encoded by y; i.e., $f_y(x) = y_x$, for any $x \in \{0,1\}^n$.

Lemma 2.1. *A circuit of size s can be specified using $O(s \log s)$ bits. Hence, there are at most $2^{O(s \log s)} = s^{O(s)}$ distinct circuits of size at most s.*

Theorem 2.2 ([102])**.** *The fraction of functions on n variables that have a circuit of size less than $2^n/(3n)$ is $o(1)$.*

We will need the following result, which says that finite field arithmetic can be performed by almost linear-size circuits.

Fact 2.3 (See, e.g., [114, 38])**.** *For any integer $\ell > 0$, let the elements in $\mathbb{F}_{2^\ell}$ be represented by ℓ-bit strings. Then, addition over $\mathbb{F}_{2^\ell}$ can be performed by a circuit of size $O(\ell)$ and multiplication over $\mathbb{F}_{2^\ell}$ can be performed by a circuit of size $\widetilde{O}(\ell)$.*

Boolean Circuits of Constant Depth

Constant-depth circuits have depth that is constant and independent of their number of inputs. These circuits are defined over the *unbounded* fan-in $\{\text{AND}, \text{OR}, \text{NOT}\}$ basis. The class of languages computed by a collection of constant-depth polynomial-size circuits is AC^0.

2.1.2 De Morgan Formulas

De Morgan formulas are Boolean circuits over the bounded fan-in $\{\text{AND}, \text{OR}\}$ basis such that their underlying graph (which is a DAG) is a binary tree with leaves in the set

$$\{0, 1, x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\},$$

where n is the number of variables. The *size* of a De Morgan formula is the number of its non-constant leaf gates. For a formula F, we denote by $L(F)$ the size of F. Let $s : \mathbb{N} \to \mathbb{N}$. We denote by $\mathsf{FORMULA}[s]$ the class of languages computed by a collection $\{F_n\}_{n\in\mathbb{N}}$ of size-$s(n)$ De Morgan formulas.

We will sometimes refer to *formulas over an arbitrary basis*. In this case, we mean that the basis can be any subset of the set of all Boolean functions from $\{0,1\}^2$ to $\{0,1\}$.

2.1.3 Branching Programs

A *branching program (BP)* is a DAG with three special vertices: A vertex s (which is called the *source*) an *accepting vertex* h_1 and a *rejecting vertex* h_0 (the *sinks*).

On input $x \in \{0,1\}^n$, the computation starts at s and follows a directed path from s to some h_b, with $b \in \{0,1\}$. On this occasion, the output of the computation is b. In each step, the computation queries some input x_i, for $i \in [n]$, and then visits some other node, depending on the value of the variable just queried, namely 0 or 1, through an edge with label "$x_i = 0$" or

"$x_i = 1$," respectively. A branching program P *computes a language* $L \subseteq \{0,1\}^n$ in a natural way, i.e., $x \in L$ if and only if, on input x, the computation path that P follows starts at s and finishes at h_1.

If the branching program is layered and the variable queried within each layer is the same, then the branching program is called *oblivious*. If the branching program queries each variable at most once, then the branching program is called a *read-once branching program (ROBP)*. If the branching program is oblivious and always queries the variables in some known order, where it is known beforehand which variable is queried at each layer, then the branching program is called *known-order*, else it is called *any-order*.

We define the *size* of a branching program to be the number of its edges.

2.2 The Minimum Circuit Size Problem

We give here a formal definition of MCSP.

Definition 2.4. The *Minimum Circuit Size Complexity Problem (*MCSP*)* is defined as follows.

- Input: A truth table $x \in \{0,1\}^{2^n}$, that encodes a Boolean function $f : \{0,1\}^n \to \{0,1\}$, and a size parameter $0 \leq \theta \leq 2^n - 1$ in binary.
- Question: Is there a circuit of size at most θ that computes f?

We will also make use of the following parameterized version of MCSP.

Definition 2.5. Let $\theta : \mathbb{N} \to \mathbb{N}$ be a function. The *Minimum Circuit Size Complexity Problem of parameter* θ *(*MCSP$[\theta]$*)* is defined as follows.

- Input: A truth table $x \in \{0,1\}^{2^n}$, that encodes a Boolean function $f : \{0,1\}^n \to \{0,1\}$.
- Question: Is there a circuit of size at most $\theta(n)$ that computes f?

2.3 Pseudorandom Generators

We will need the following fundamental definition of pseudorandom generators.

Definition 2.6 (Pseudorandom generators)**.** Let $s : \mathbb{N} \to \mathbb{N}$ be a function such that $s(n) < n$ for all $n \in \mathbb{N}$, $\mathcal{F}$ a class of Boolean functions, and $\varepsilon : \mathbb{N} \to (0, 1)$. A *pseudorandom generator (PRG) of seed length s that ε-fools $\mathcal{F}$* is a function $\left\{G_n : \{0,1\}^{s(n)} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$ such that for every function $f = \{f_n\}_{n\in\mathbb{N}} \in \mathcal{F}$ it is the case that

$$\left| \mathop{\mathbf{Pr}}_{z\sim\{0,1\}^{s(n)}}[f_n(G_n(z)) = 1] - \mathop{\mathbf{Pr}}_{x\sim\{0,1\}^n}[f_n(x) = 1] \right| \leq \varepsilon(n)\,.$$

2.4 Useful Facts About Distributions

2.4.1 k-wise Independent Distributions

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

Definition 2.7 (k-wise independence)**.** A distribution $X = (X_1, \ldots, X_n)$ over $[m]^n$ is called *k-wise independent* if the distributions $X_1, \ldots, X_n$ over $[m]$ are independent and identically distributed and, moreover, $\mathbf{Pr}[X_i = b] = 1/m$ for every $i \in [n]$ and $b \in [m]$. If $k = 2$, then a k-wise independent distribution is called *pairwise independent*.

Note that, according to Definition 2.7, if $X = (X_1, \ldots, X_n)$ is a k-wise independent distribution over $[m]^n$, then for every $1 \leq i_1 < \cdots < i_k \leq n$ and every $b_1, \ldots, b_k \in [m]$, we have

$$\mathbf{Pr}[X_{i_1} = b_1, \ldots, X_{i_k} = b_k] = \prod_{j=1}^{k} \mathbf{Pr}\big[X_{i_j} = b_j\big] = \frac{1}{m^k}.$$

2.4.2 Almost Linear-size k-wise Independent Generators

A *k-wise independent generator* is a function from binary strings to binary strings that takes as input a random seed and stretches that seed to a string that follows a k-wise independent distribution.

We will use efficient constructions for k-wise independent generators.

Lemma 2.8. *For any integer $k > 0$, there exists a k-wise independent generator $G : \{0,1\}^r \to [m]^n$, with $r = k \cdot \max\{\log n, \log m\}$, such that the following holds. There exists a circuit of size*

$$k \cdot \max\{\widetilde{O}(\log n), \widetilde{O}(\log m)\}$$

such that, given $j \in \{0,1\}^{\log n}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th coordinate of $G(z)$.

Proof. Let $n' = \max\{n, m\}$ and suppose $n' = 2^\ell$. We view the elements in $\mathbb{F}_{n'}$ as ℓ-bit strings. Consider the following function $g : \mathbb{F}_{n'} \times \mathbb{F}_{n'}^k \to \mathbb{F}_{n'}$ given by

$$g(i, z_0, \ldots, z_{k-1}) = z_0 + z_1 \cdot i + \cdots + z_{k-1} \cdot i^{k-1}.$$

It is known (see [110, Proposition 3.33]) that the function $G : \mathbb{F}_{n'}^k \to \mathbb{F}_{n'}^{n'}$ defined as

$$G(z_0, \ldots, z_{k-1}) = \big(g(1, z_0, \ldots, z_{k-1}), \ldots, g(n', z_0, \ldots, z_{k-1})\big),$$

is a k-wise independent generator.

Using Fact 2.3 it is straightforward to implement a circuit of size $k \cdot \widetilde{O}(\ell)$ that computes $g(j, z)$. Note that to get an output in $[m]$ we can simply output the first $\log m$ bits of $G(z)_j$, since the field has characteristic 2. □

2.4.3 δ-biased Distributions

Next, we define δ-biased distributions.

Definition 2.9. Let $n \in \mathbb{N}$ and $0 < \delta < 1$. A distribution D over $\{0,1\}^n$ is said to be *δ-biased* if, for any $S \subseteq [n]$, it is the case that

$$\left| \Pr_{X \sim D} \left[\prod_{i \in S} (-1)^{x_i} = 1 \right] - \Pr_{X \sim \{0,1\}^n} \left[\prod_{i \in S} (-1)^{x_i} = -1 \right] \right| \leq \delta,$$

where $X = (x_1, \ldots, x_n)$.

2.4.4 Almost Linear-size δ-biased Generators

A *δ-biased generator* is a function $G : \{0,1\}^s \to \{0,1\}^n$ that takes as input a random seed $z \in \{0,1\}^s$ and its output $G(z) \in \{0,1\}^n$ follows a δ-biased distribution over $\{0,1\}^n$ (see Definition 2.9).

We will use efficient constructions for δ-biased independent generators. In Theorem 2.10 we observe that the δ-biased distribution of Alon, Goldreich, Håstad and Peralta [13] encodes truth tables of Boolean functions that have low circuit complexity. Our contribution here lies exactly at this observation.

Theorem 2.10. *There exists a δ-biased generator $G : \{0,1\}^s \to \{0,1\}^n$ with seed length $s = O(\log(n/\delta))$ such that its output strings encode Boolean functions of circuit complexity $\widetilde{O}(\log(n/\delta)) \cdot \mathrm{polylog}(n)$.*

Proof. We use the standard construction of an δ-biased generator G_0 of [13]. Let $m := \log(n/\delta)$, and take a field of size 2^m. For random seeds $a, b \in \mathbb{F}_{2^m}$ of length $2m$, the ith bit of $G_0(a,b)$ is defined as $\langle a^i, b \rangle$, i.e., the inner product of the binary representations of a^i and b. It was shown in [13] that, for a, b chosen uniformly at random from $\mathbb{F}_{2^m}$, the distribution of $G_0(a,b)$ is δ-biased.

We claim that, for every $a, b \in \mathbb{F}_{2^m}$, there exists a circuit of size $O(\log(n/\delta) \cdot \log n)$ that takes $i \in [n]$ as an input of length $\log n$ and computes $\langle a^i, b \rangle$. Indeed, we hardwire a^{2^j} for all $j \leq \log n$ into a circuit; given an input $i \in [n]$, one can compute a^i by multiplying a^{2^j} for all j such that the j-th bit of the binary representation of i is 1. This can be done with a circuit of size $\widetilde{O}(m) \cdot \log n$ by using Fact 2.3. It remains to compute the inner product of a^i and b, which can be done with a linear-size circuit. Overall, we obtain a circuit of size $\widetilde{O}(m) \cdot \log n = \widetilde{O}(\log(n/\delta)) \cdot \log n$. □

2.5 Restrictions and Selections

A restriction for an n-variate Boolean function f, usually denoted as $\rho \in \{0, 1, *\}^n$, specifies a way of fixing the values of some subset of variables for f. That is, if ρ_i is $*$, we leave the i-th variable unrestricted and otherwise fix its value to be $\rho_i \in \{0, 1\}$. We denote by $f|_\rho$ the restricted function after the variables are restricted according to ρ, and denote by $\rho^{-1}(*)$ the set of unrestricted variables. A random restriction is then a distribution over restrictions. We will often view sampling a random restriction as a two-step process: The first step is selecting (in some random manner) a subset (that is, a *selection*) of unrestricted variables (also called the "star" or "$*$" variables) and the second step is fixing (in some random manner) the values of all the other variables. Then, a random restriction over n variables can also be perceived as distribution over pairs $(\sigma, \beta) \in \{0, 1\}^n \times \{0, 1\}^n$, where σ (as a characteristic string) specifies the set of unrestricted variables, and β specifies the values for fixing the restricted variables.

We say that a random restriction (or random selection) is p-regular if each variable is left unrestricted with probability p. One way to generate a p-regular random restriction is to leave each variable, independently, unrestricted with probability p, and otherwise assign to it a 0 or a

1, uniformly at random. Such a random restriction is called a *(truly) p-random restriction.* Note that to sample such a restriction, we can first pick a string in $\{0,1\}^{n\cdot\log(1/p)} \cong [1/p]^n$ to specify the selection of the unrestricted variables, where a coordinate is unrestricted if and only if all of its corresponding $\log(1/p)$ bits are 0, and then a string in $\{0,1\}^n$ to specify the values assigned to each of the restricted variables. So sampling a restriction in this way requires $n \cdot \log(1/p) + n$ random bits. We can also generate a restriction in a pseudorandom manner, which may use fewer random bits. For example, one way to do this is to use a limited-independence distribution, so that each variable is set to be unrestricted with probability p, and any k of the variables are independent. Note that such a "pseudorandom selection" can be obtained using a k-wise independent distribution on $[1/p]^n$. Also, we can let each variable be assigned a 0 or a 1 uniformly at random in a way such that any k of the variables are independent; this again can be done using a k-wise independent distribution on $\{0,1\}^n$.

The above discussion implicitly assumes that $1/p$ is a power of 2. If $1/p$ is not a power of 2, then the reader may assume that the quantity $1/p$ is replaced by the smallest power of 2 that is at least $1/p$ (and therefore at most $2/p$). This may result in sampling more bits than needed; in this case, we may only use as many bits as we need and discard the rest.

Finally, note that we can also get a restriction by combining a sequence of restrictions $\rho_1, \ldots, \rho_t$, in a natural way, namely by applying the sub-restrictions one by one. In this case, we write the final restriction as $\rho_1 \circ \cdots \circ \rho_t$.

2.6 Kolmogorov Complexity and Variants

2.6.1 A Universal Turing Machine

In what follows, we fix some universal Turing machine (UTM) U. Let $y, \Pi, z \in \{0,1\}^*$ and $t \in \mathbb{N}$. The notation $U^{\Pi,y}\big(z, 1^t\big)$ denotes the output of U when U runs the program Π on input z for at most t steps, given that U has oracle access to y. The notation $U^{\Pi,y}(z)$ denotes the output of U when U runs the program Π on input z, until Π halts (if this is the case, otherwise Π runs forever), whereby U has oracle access to y. In this work, we will assume that the output of any UTM considered is either 1 or 0, on any of their inputs.

2.6.2 Kolmogorov Complexity

Given strings $x, y \in \{0,1\}^*$, we define the *Kolmogorov complexity of x given y*, denoted $\mathrm{K}(x \mid y)$, to be the size $|\Pi|$ of a minimum-size program $\Pi \in \{0,1\}^*$ such that for all $1 \leq i \leq |x|$ it is the case that $U^{\Pi,y}(i) = x_i$. For all strings $x \in \{0,1\}^*$, we define $\mathrm{K}(x)$ to be equal to $\mathrm{K}(x \mid \lambda)$.

2.6.3 KT Complexity

Given strings $x, y \in \{0,1\}^*$, we define the *KT complexity of x given y*, denoted $\mathrm{KT}(x \mid y)$, to be the minimum value of $|\Pi| + t$ over programs $\Pi \in \{0,1\}^*$ and run-time bounds $t \in \mathbb{N}$ whereby for all $1 \leq i \leq |x|$ it is the case that $U^{\Pi,y}\big(i, 1^t\big) = x_i$. For all strings $x \in \{0,1\}^*$, we define $\mathrm{KT}(x)$ to be equal to $\mathrm{KT}(x \mid \lambda)$.

The following result gives a trivial upper bound on the KT complexity of a string.

Lemma 2.11 ([6])**.** *There is a $c > 0$ such that for all $x \in \{0,1\}^*$ it is the case that $\mathrm{KT}(x)$ is at most $|x| + c \log |x|$.*

Chapter 3

MCSP Lower Bounds From Local PRGs

As we saw in Section 2.3 (Definition 2.6), for a class $\mathfrak{C}$ of N-variate Boolean functions, a PRG that fools $\mathfrak{C}$ is a function G mapping short binary strings (seeds) to longer binary strings so that every function in $\mathfrak{C}$ accepts G's output on a uniformly random seed with about the same probability as that for an actual uniformly random string. In this chapter we will explain our main method for obtaining lower bounds for MCSP by making use of local pseudorandom generators (local PRGs).

We say that a PRG is *local* if its N-bit output (viewed as the truth table of some function) has small circuit complexity, for any of its seeds. By the definition of circuit complexity (Section 2.1.1), this implies that, for any fixed seed to the PRG, there exists a small circuit such that given $j \in [N]$ as input the circuit computes the j-th bit of the output of the PRG output (where the size of the circuit is measured relative to its input length, namely $\log N$). Note that our notion of local PRGs does not require that the PRG in question is explicit; that

is, we do not require that a local PRG can be computed by some uniform algorithm.

Local PRGs in the context of MCSP (and related problems) have been studied in previous works (see, e.g., [6, 87, 48, 47]). In this thesis, we refine the previous approaches and obtain stronger circuit lower bounds by establishing locality properties of certain PRG constructions.[1]

Suppose we have a local PRG that fools some class of circuits $\mathfrak{C}$ of size s, and we want to show that MCSP cannot be computed by any size-s circuit in $\mathfrak{C}$. Suppose some size-s circuit C in $\mathfrak{C}$ computes MCSP. Using the fact that a random function has almost maximum circuit complexity, we have that C will output FALSE on most of its inputs (by setting the size parameter θ to be a non-trivial quantity that is asymptotically smaller than $2^n/n$, where n is the input length of the function). If we replace the uniformly random inputs with the outputs of the local PRG, then, by the definition of PRG, C will still output FALSE with a large probability. However, since the PRG is local, all of its outputs have circuit complexity smaller than the size parameter θ, and hence must be accepted by C. A contradiction.

To get a strong lower bound, we would like to make the above argument to work for large s. Note that the local complexity of the PRG, λ, is a function of the size s of the circuit C (that has N inputs), and we need this local complexity to be "non-trivial" in order to reach a contradiction. Therefore, we want to choose s so that this local complexity remains asymptotically smaller than $2^n/n$. As a result, the final lower bound (i.e., the largest s that we can choose) is determined by the local complexity λ. So one of the main questions we study in this thesis is this:

[1] Note that the notion of a local PRG can be also found in the context of cryptography [31], where a PRG $G : \{0,1\}^n \to \{0,1\}^m$ is called *k-local*, for some constant $k > 0$, if every output PRG bit $G(x)_j$, for any $x \in \{0,1\}^n$ and $j \in [m]$, depends only on k input bits $x_{i_1}, \ldots, x_{i_k}$, for $i_1, \ldots, i_k \in [n]$. In our work, however, locality refers to the circuit complexity of the PRG at hand and the output bits of our PRGs may depend on a super-constant number of input bits.

What is the smallest local complexity of a PRG that fools a given class?

We formally define local PRGs in Section 3.1, and then proceed to explain the relation between local PRGs and MCSP lower bounds in Section 3.2.

3.1 Local Functions and PRGs

Here we formally define local functions (and local PRGs).

Definition 3.1 (Local functions)**.** Let $r, \lambda : \mathbb{N} \to \mathbb{N}$ be functions such that $r(N) < N$ for all $N \in \mathbb{N}$. A function $\left\{G_N : \{0,1\}^{r(N)} \to \{0,1\}^N\right\}_{N\in\mathbb{N}}$ is $\lambda(N)$-*local* if for every $N \in \mathbb{N}$ and every string $z \in \{0,1\}^{r(N)}$ it is the case that $\mathrm{CC}(G_N(z)) \leq \lambda(N)$. We will be referring to λ as the *locality* or the *local complexity* of G.

According to Definition 3.1, a λ-*local PRG* is a PRG that is λ-local.

Remark 3.2. In this work, the pseudorandom constructions $\left\{G_N : \{0,1\}^{r(N)} \to \{0,1\}^N\right\}_{N\in\mathbb{N}}$ that we will encounter will be $\lambda(N)$-local for $\lambda(N) \leq \widetilde{O}(r(N))\,\mathrm{polylog}(N)$, for all $N \in \mathbb{N}$ (as per the notation of Definition 3.1).

Remark 3.3. Our notion of local PRGs resembles that of *succinct pseudorandom distributions*, put forward by Santhanam [98]. In particular, our notion of a λ-local PRG $G_N : \{0,1\}^{r(N)} \to \{0,1\}^N$ that ε-fools a class of functions $\mathcal{F}$ (notation here is as in Definition 3.1 and Definition 2.6) yields a $\mathsf{SIZE}[\lambda(N)]$-succinct pseudorandom distribution $G_N(z)$ (where $z \sim \{0,1\}^{r(N)}$) against $\mathcal{F}$ with error ε.

3.2 MCSP Lower Bounds From Local PRGs: Our Framework

We describe how to use local PRGs to obtain circuit lower bounds for MCSP. Note how we only consider the case where the PRG at hand fools some *deterministic* model of computation. (See Theorem 8.3 for an adaptation of Theorem 3.4 to the case of *randomized* models of computation.)

Theorem 3.4. *For any (deterministic) computational model $\mathcal{M}$, let $s : \mathbb{N} \to \mathbb{N}$ be such that* $\mathrm{MCSP}[N/(100 \log N)]$ *on truth tables of length N can be computed by a device of size $s(N)$ in $\mathcal{M}$. If for a function $\lambda : \mathbb{N} \to \mathbb{N}$ there exists a $\lambda(N)$-local PRG that $(1/3)$-fools size-$s(N)$ devices from $\mathcal{M}$, then $\lambda(N) \geq \frac{N}{100 \log N}$.*

Proof. Let C be a size-$s(N)$ device in $\mathcal{M}$ such that C computes $\mathrm{MCSP}[N/(100 \log N)]$ on truth tables of length N. Let $r : \mathbb{N} \to \mathbb{N}$ and $G : \{0,1\}^{r(N)} \to \{0,1\}^N$ be a $\lambda(N)$-local PRG that fools C.

For the sake of contradiction, suppose that

$$\lambda(N) < \frac{N}{100 \log N}.$$

On the one hand, since most functions require circuits of size greater than $\frac{N}{3 \log N} \geq \frac{N}{100 \log N}$ (Theorem 2.2) and C computes MCSP, we have

$$\mu := \mathop{\mathbf{Pr}}_{x \sim \{0,1\}^N}[C(x) = 0] \geq 1/2.$$

Also, since G $(1/3)$-fools C, we have

$$\mathop{\mathbf{Pr}}_{z \sim \{0,1\}^{r(N)}}[C(G(z)) = 0] \geq \mu - 1/3 \geq 1/6.$$

On the other hand, because G is $\lambda(N)$-local, we must have $C(G(z)) = 1$ for every z. This gives a contradiction. □

We will use Theorem 3.4 to obtain lower bounds for parameterized MCSP, where its size parameter is $N/(100 \log N) = 2^{\Omega(n)}$, by proving its contrapositive statement. That is, by proving that there exists a λ-local PRG for $\lambda(N) < N/(100 \log N)$ that fools a model of computation $\mathcal{M}$, one can derive an $\mathrm{MCSP}\left[2^{\Omega(n)}\right]$ lower bound against $\mathcal{M}$ by invoking the contrapositive of Theorem 3.4.

Notes

The observations of this chapter are products of joint work with Mahdi Cheraghchi, Valentine Kabanets, and Zhenjian Lu [29], and (to some extent) Mahdi Cheraghchi, Shuichi Hirahara, and Yuichi Yoshida [28].

It is a straightforward observation that a local hitting set generator (HSG) is sufficient for (a version of) Theorem 3.4 to work. (Note that a HSG is a notion that is weaker than that of a PRG.)

Chapter 4

MCSP Lower Bounds Against Formulas and Branching Programs

For many years after the modern study of MCSP was initiated by Kabanets and Cai [62], there were no known MCSP lower bounds against De Morgan formulas or branching programs. Indeed, the first MCSP lower bound against De Morgan formulas was proved only in 2017, by Hirahara and Santhanam [48].

In this chapter, we improve the state-of-the-art and show that computing MCSP on truth tables of length N requires $N^{3-o(1)}$-size De Morgan formulas (see Theorem 1.1), improving the recent $N^{2-o(1)}$ lower bound by Hirahara and Santhanam [48], and $N^{2-o(1)}$-size formulas over an arbitrary basis or general branching programs (see Theorem 1.2; no non-trivial lower bound was known for MCSP against these models).

Our formula lower bound for MCSP is obtained by applying the framework of Chapter 3 to a local PRG that fools formulas. The state-of-the-art PRG that fools formulas is given by Impagliazzo, Meka, and Zuckerman [57] , which we refer to as the IMZ PRG. Their PRG has a

seed length of $s^{1/3+o(1)}$ for size s formulas (note that such a PRG is useful against sub-cubic formulas only). If we want to utilize the IMZ PRG to get an MCSP lower bound against formulas, we will need to argue that the IMZ PRG is local.

In fact, in order to get an almost-cubic lower bound, it would suffice to have such a PRG that is local in the sense of Definition 3.1, that is, any single output bit of the PRG (on any given fixed seed) can be computed by a circuit of size comparable to its seed length, which is $s^{1/3+o(1)}$. However, by inspecting the construction, the IMZ PRG does not seem to have such a property, and a straightforward implementation seems to require a circuit of size at least $s^{2/3}$, which yields a weaker lower bound for MCSP.

To overcome this issue, we present an alternative PRG useful against sub-cubic formulas which is local. The construction of this PRG can be viewed as a modification of the IMZ PRG. At a high level, it is based on the Ajtai-Wigderson construction [3], which is a framework for constructing PRGs against computations that can be simplified under (pseudo)random restrictions.

The Ajtai-Wigderson framework is then combined with the ideas for reducing (recycling) random bits using an extractor, by exploiting communication bottlenecks in computations [84]. Our modification, particularly the utilization of the Ajtai-Wigderson construction, allows us to compute any output bit of the PRG efficiently by reducing the number of calls to the extractor. Using some crucial observations on the circuit complexity of certain pseudorandom objects, we get a PRG that is locally computable by a $s^{1/3+o(1)}$-size circuit.

Remark 4.1. As mentioned above, the Ajtai-Wigderson framework [3] is very useful here because it shows a way of producing n pseudorandom bits in *stages*. Consider a partition $\{S_i\}_{i=1}^{k}$ of $[n]$. Ajtai and Wigderson put forward the idea of constructing n pseudorandom bits, viewed as values of Boolean variables $x_1, \ldots, x_n$, by first producing bits for the variables x_i $(i \in S_1)$ that

have their indices in S_1, then producing bits for the variables x_i ($i \in S_2$) that have their indices in S_2, etc., until the variables with indices in S_k are produced. We employ this iterative idea here, because it gives an efficient way of computing the output bits of our PRG (see Lemma 4.7 and its proof). This is made apparent in Figure 4.1, whereby the sets $S := \{S_i\}_{i=1}^{t}$ form a partition of $[N]$, and each output bit of G depends only on *one* of the sets in S.

The MCSP lower bounds against formulas over an arbitrary basis or branching programs are obtained similarly to those for De Morgan formulas. The idea is to construct local PRGs against these models by modifying the PRGs in [57]. Then, by applying our framework, we get the desired lower bounds.

We give the necessary background in Section 4.1. We prove the almost-cubic De Morgan formula lower bound for MCSP (Theorem 1.1) in Section 4.2, and the almost-quadratic lower bounds against formulas over an arbitrary basis or branching programs (Theorem 1.2) in Section 4.3.

4.1 Preliminaries

4.1.1 Probability Theory

We will need the following concentration bound for k-wise independent distributions, which is an application of Cantelli's inequality.

Proposition 4.2. *For any $0 < p < 1$, let $X_1, \ldots, X_n$ be pairwise independent random variables over $\{0,1\}$ such that $\mathbf{Pr}[X_i = 1] = p$ for all $i \in [n]$. Let $X := \sum_{i=1}^{n} X_i$. Then, it is the case that*

$$\mathbf{Pr}[X \leq \mathbf{E}[X]/2] \leq \frac{4}{\mathbf{E}[X]}.$$

The following simple fact will be convenient for us.

Lemma 4.3. *Let X and Y be two random variables that take values in $\{0,1\}$ and $\mathcal{E}$ be some event. If*

- $|\mathbf{E}[X \mid \mathcal{E}] - \mathbf{E}[Y \mid \mathcal{E}]| \leq \varepsilon_1$ *and*
- $\mathbf{Pr}[\neg\mathcal{E}] \leq \varepsilon_2$,

then $|\mathbf{E}[X] - \mathbf{E}[Y]| \leq \varepsilon_1 + \varepsilon_2$.

Proof. We have

$$\mathbf{E}[X] = \mathbf{E}[X \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[X \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}],$$

and

$$\mathbf{E}[Y] = \mathbf{E}[Y \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{E}] + \mathbf{E}[Y \mid \neg\mathcal{E}] \cdot \mathbf{Pr}[\neg\mathcal{E}].$$

Then,

$$\begin{aligned}\mathbf{E}[X] - \mathbf{E}[Y] &= (\mathbf{E}[X \mid \mathcal{E}] - \mathbf{E}[Y \mid \mathcal{E}]) \cdot \mathbf{Pr}[\mathcal{E}] + (\mathbf{E}[X \mid \neg\mathcal{E}] - \mathbf{E}[Y \mid \neg\mathcal{E}]) \cdot \mathbf{Pr}[\neg\mathcal{E}] \\ &\leq |\mathbf{E}[X \mid \mathcal{E}] - \mathbf{E}[Y \mid \mathcal{E}]| + \mathbf{Pr}[\neg\mathcal{E}] \\ &\leq \varepsilon_1 + \varepsilon_2.\end{aligned}$$

The fact $\mathbf{E}[Y] - \mathbf{E}[X] \leq \varepsilon_1 + \varepsilon_2$ can be similarly shown. □

4.1.2 Circuit Complexity

We will make use of the following facts about Boolean circuits.

Lemma 4.4. *For any integer $t > 0$, there exists a circuit C of size $\widetilde{O}(t)$ such that, given any string $x \in \{0,1\}^t$, the circuit does the following:*

- *If $x = 0^t$, then C outputs $(0, 0^{\log t})$.*
- *If $x \neq 0^t$, then C outputs $(1, q)$, where $q \in \{0,1\}^{\log t}$ is the index of the first bit in x that is not 0.*

Proof. Define $z^{(0)} = \left(0, 0^{\log t}\right)$ and $z^{(i)}$, for any $i = 1, \ldots, t$, recursively as follows:

$$z^{(i)} = \begin{cases} z^{(i-1)}, & \text{if } (z^{(i-1)})_1 = 1, \\ z^{(i-1)}, & \text{if } (z^{(i-1)})_1 = 0 \text{ and } x_i = 0, \text{ and} \\ (1, i), & \text{if } (z^{(i-1)})_1 = 0 \text{ and } x_i = 1. \end{cases}$$

Note that each $z^{(i)}$ can be computed in polylog(t) size given $z^{(i-1)}$. Using a circuit of size $\widetilde{O}(t)$ we can compute $z^{(t)}$, which is our output. □

The following circuit upper bound for the addressing (storage access) function is well-known (see, e.g., [116]); we include a proof for completeness.

Lemma 4.5. *For any integers $t, m > 0$, there exists a circuit of size $O(t \cdot m)$ such that, given any string $y = (y_1, \ldots, y_t)$, where $y_i \in \{0,1\}^m$, for each i, and an index $i \in \{0,1\}^{\log t}$, the circuit outputs y_i.*

Proof. We first look at the first bit (i.e., the least significant bit in binary) of i and output either the first half of y (i.e., $y_1, \ldots, y_{t/2}$), if the first bit is 0, or the second half (i.e., $y_{(t/2)+1}, \ldots, y_t$), if the first bit is 1; denote this output as $y^{(1)}$. This can be done by a circuit of size $c \cdot t \cdot m$, for some constant $c > 0$. Then, we look at the second bit of i and output either the first half or the second half of $y^{(1)}$, denoted as $y^{(2)}$. This can be done by a circuit of size $c \cdot t \cdot m/2$. We repeat the above process $\log t$ times, in total, until we get $y^{(\log t)}$, which is y_i. The circuit complexity of this procedure is

$$c \sum_{k=1}^{\log t} \frac{tm}{2^{k-1}} = O(tm)\,.$$

□

4.2 Almost-cubic De Morgan Formula Lower Bounds for MCSP

In this section, we present our almost-cubic De Morgan formula lower bound for MCSP.

Theorem 4.6 (Theorem 1.1, restated)**.** *Any De Morgan formula computing* $\mathrm{MCSP}\left[2^{\Omega(n)}\right]$ *on truth tables of length* $N := 2^n$ *must have size at least* $N^3/2^{O\left(\log^{2/3} N\right)}$.

We will construct a local PRG that fools sub-cubic formulas. That is, given as input an index j, the j-th bit of the PRG can be computed by a circuit of size that is comparable to its seed length, which in our case is around $s^{1/3}$ for size s formulas.

Lemma 4.7. *For any* $s \geq N$*, there exists a* $s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$*-local PRG of seed length* $s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$ *that* $(1/3)$*-fools size-*s *De Morgan formulas on* N *inputs.*

Given the local PRG in Lemma 4.7, we can combine it with our Theorem 3.4 to obtain a formula lower bound for MCSP.

Proof of Theorem 4.6. Let $s \leq N^3$ be such that $\mathrm{MCSP}\left[2^{\Omega(n)}\right]$ on truth tables of length $N := 2^n$ can be computed by some formula of size s. By Theorem 3.4 and Lemma 4.7, we have

$$s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)} \geq N/(100 \log N);$$

then,

$$s \geq N^3 / \left(2^{O\left(\log^{2/3} N\right)} 100^3 \log^3 N\right) = N^3/2^{O\left(\log^{2/3} N\right)}. \qquad \square$$

The rest of this section is devoted to proving Lemma 4.7.

4.2.1 Almost Linear-size Extractors

Our PRG will make use of randomness extractors. Here, we describe an extractor that is computable by a circuit of size that is almost linear in the length of its input. We start by reviewing some basic definitions regarding extractors.

Definition 4.8 (ε-closeness and statistical distance)**.** Let $0 \leq \varepsilon \leq 1$. We say two distributions X and Y (over some universe D) are *ε-close* if their *statistical distance*, defined as

$$\max_{T:D\to\{0,1\}} \left|\mathbf{Pr}[T(X)=1] - \mathbf{Pr}[T(Y)=1]\right|,$$

is at most ε.

Definition 4.9 (Min-entropy)**.** Let X be a random variable. The *min-entropy* of X, denoted by $H_\infty(X)$, is the largest real number k such that $\mathbf{Pr}[X=x] \leq 2^{-k}$ for every x in the support of X. If X is a distribution over $\{0,1\}^{\widetilde{N}}$ with $H_\infty(X) \geq k$, then X is called a $\left(\widetilde{N}, k\right)$*-source.*

Definition 4.10 (Extractors)**.** A function $E : \{0,1\}^{\widetilde{N}} \times \{0,1\}^d \to \{0,1\}^m$ is an (k,ε)*-extractor* if, for any $\left(\widetilde{N}, k\right)$-source X, the distribution $E(X, U_d)$ is ε-close to U_m.

We now state the extractor, which for a high min-entropy source extracts a constant fraction of the min-entropy, using seeds of polylogarithmic length.

Lemma 4.11 (Almost-linear-size extractors, following [84])**.** *There exists some randomness extractor $E : \{0,1\}^{\widetilde{N}} \times \{0,1\}^d \to \{0,1\}^m$ that is an $(\widetilde{N}/2, \varepsilon)$-extractor with $m = \Omega(\widetilde{N})$ and $d = \mathrm{polylog}(\widetilde{N}/\varepsilon)$. Moreover, E can be computed by a circuit of size $\widetilde{N} \cdot \mathrm{polylog}(\widetilde{N}/\varepsilon)$.*

Proof of Lemma 4.11

In proving Lemma 4.11, we start with some definitions. The extractor works for sources of high min-entropy.

Definition 4.12 (Dense source)**.** We say that a distribution over $\{0,1\}^{\widetilde{N}}$ is a *δ-source* if it has min-entropy at least $\delta \cdot \widetilde{N}$.

Definition 4.13 (Block-wise source)**.** A distribution $X = (X_1, \ldots, X_s)$ over $\{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_s}$ is called a *block-wise δ-source* if, for every $x_1, \ldots, x_{i-1}$, $X_i|_{X_1=x_1,\ldots,X_{i-1}=x_{i-1}}$ is a δ-source (i.e., has min-entropy at least $\delta \cdot \ell_i$).

The extractor will make use of universal hashing, which we define below.

Definition 4.14 (k-wise independent hashing)**.** A family of hash functions $\mathcal{H} = \{h : \{0,1\}^{\widetilde{N}} \to \{0,1\}^m\}$ is called *k-wise independent* if, for any $x_1, \ldots, x_k \in \{0,1\}^{\widetilde{N}}$, where $x_1, \ldots, x_k$ are distinct, and $y_1, \ldots, y_k \in \{0,1\}^m$, we have

$$\Pr_{h \sim \mathcal{H}}[h(x_1) = y_1 \wedge \cdots \wedge h(x_k) = y_k] = (1/2^m)^k .$$

$\mathcal{H}$ is also called a *universal hash family* if it is 2-wise independent.

It is straightforward to see that any k-wise independent hashing family can be defined using some k-wise independent distribution. As a result, by Lemma 2.8, we have the following construction of k-wise independent hash families.

Lemma 4.15. *There exists a k-wise independent hash family $\mathcal{H} = \{h : \{0,1\}^{\widetilde{N}} \to \{0,1\}^m\}$ such that, given any $h \in \mathcal{H}$, as a kn-bit string, the function h can be computed by a circuit of size $k \cdot \widetilde{O}(\max\{\widetilde{N}, m\})$.*

The Nisan-Zuckerman extractor consists of two parts. The first part, block-wise source conversion, takes the source of high min-entropy and converts it into an almost block-wise source by building a list of "blocks." The second part, block-wise source extraction, takes the resulting block-wise source of the previous part and extracts the randomness "block-by-block," using some hash-based extractor. Next, we describe some basic component functions as well as how they are combined to perform the respective task of each part. The main focus here is around the circuit complexity of these procedures and we will not get into details about their correctness. Interested readers are referred to [84, Section 5] for details on the correctness.

In the following, we only work with δ-sources and block-wise δ'-sources where δ and δ' are constants.

Block-wise source converter D. This function has the following parameters:

- $\widetilde{N}$, the size of the original input;
- δ, the quality of the input source;
- $\ell_1 \leq \cdots \leq \ell_s \leq \widetilde{N}$, the size of each block; and
- k, the amount of independence used.

We first describe how to build one block using a function that we call B. To build the i-th block, on input $x \in \{0,1\}^{\widetilde{N}}$ and $y_i \in \{0,1\}^{k \log \widetilde{N}}$, the function B first divides x into ℓ_i contiguous disjoint sets $A_1, \ldots, A_{\ell_i}$, each of size $m_i = \widetilde{N}/\ell_i$. It then uses the $(k \log \widetilde{N})$-bit string y_i to pick, k-wise independently, $j_1, \ldots, j_{\ell_i}$, where $j_q \in [m_i]$, for each $q \in [\ell_i]$, and outputs the ℓ_i-bit vector

$$\left((A_1)_{j_1}, \ldots, (A_{\ell_i})_{j_{\ell_i}}\right).$$

The block-wise source converter D works as follows.

1. **Input:** $x \in \{0,1\}^{\widetilde{N}}$ and $y_1, \ldots, y_s \in \{0,1\}^{k \log \widetilde{N}}$.
2. **Output:** $(B(x, y_1), \ldots, B(x, y_s)) \in \{0,1\}^{\ell_1} \times \cdots \times \{0,1\}^{\ell_s}$.

Nisan and Zuckerman [84] showed that if the input x is from a δ-source and $k = O(\log(1/\varepsilon))$, then, for all but at most a $\varepsilon/4$ fraction of the seeds $y_1, \ldots, y_s$, the output of the function D is $(\varepsilon/4)$-close to a block-wise δ'-source, where $\delta' = \Omega(\delta/\log(1/\delta))$.

Claim 4.16. The function D can be computed using a circuit of size $s \cdot k \cdot \widetilde{O}(\widetilde{N})$.

Proof. It is sufficient to show that outputting the i-th block takes a circuit of size $k \cdot \widetilde{O}(\widetilde{N})$. On input $y_i \in \{0,1\}^{k \cdot \log \widetilde{N}}$, we can compute, using Lemma 2.8, $(j_1, \ldots, j_{\ell_i}) \in [m_i]^{\ell_i}$ with a circuit

of size

$$\ell_i \cdot k \cdot \widetilde{O}(\log(m_i \cdot \ell_i)) = k \cdot \widetilde{O}(\widetilde{N}).$$

Then, for each index j_q, with $q \in [\ell_i]$, we can compute $(A_q)_{j_q}$ using a circuit of size $O(m_i)$ (by Lemma 4.5). □

Block-wise source extractor C. This function has $s+1$ parameters:

- δ', the quality of the block source, and
- $\ell_1, \ldots, \ell_s$, the block sizes. Here, $\frac{\ell_{i-1}}{\ell_i} = 1 + \frac{\delta'}{4}$, for all $1 < i \leq s$.

The way the block-wise source extractor C works is described below.

1. **Input:** $x_1 \in \{0,1\}^{\ell_1}, \ldots, x_s \in \{0,1\}^{\ell_s}$ and $y_0 \in \{0,1\}^{2\ell_s}$.
2. For each i, we consider a universal family of hash functions,

   $$\mathcal{H}_i = \left\{ h : \{0,1\}^{\ell_i} \to \{0,1\}^{\delta' \ell_i / 2} \right\}_h,$$

   given by Lemma 4.15, and each function in $\mathcal{H}_i$ is by $2\ell_i$ bits.
3. $h_s \leftarrow y_0$.
4. For $i \leftarrow s$ down to 1: $h_{i-1} \leftarrow h_i \circ h_i(x_i)$, where "$\circ$" denotes string concatenation.
5. **Output:** h_0, excluding the bits in h_s. Note that this output is a string in $\{0,1\}^m$.

It was shown in [84] that if $x_1, \ldots, x_s$ are chosen from a block-wise δ'-source and y_0 is uniform, then the output of the function C is $\left(2 \cdot 2^{-\delta' \ell_s / 4}\right)$-close to uniform.

Claim 4.17. The function C can be computed using a circuit of size $s \cdot \widetilde{O}(\ell_1)$.

Proof. Note that, given $h_i \in \{0,1\}^{2\ell_i}$ and $x_i \in \{0,1\}^{\ell_i}$, we can compute $h_i(x_i)$ using a circuit of size $\widetilde{O}(\ell_i)$ (by Lemma 4.15). Then, to compute h_0, we need to compute h_i for $i = s-1, \ldots, 0$, which takes a circuit of size

$$\sum_{i=1}^{s} \widetilde{O}(\ell_i).$$

The above is at most $s \cdot \widetilde{O}(\ell_1)$, since ℓ_1 is the largest among $\ell_1, \ldots, \ell_s$. □

The final extractor E. The parameters are:

- $\widetilde{N}$, the size of the input source;
- δ, where $1/\widetilde{N} \leq \delta \leq 1/2$, the quality of the input source;
- ε, where $2^{-\delta\widetilde{N}} \leq \varepsilon \leq 1/\widetilde{N}$, the quality of the output distribution;
- $\delta' = \Theta(\delta/\log(1/\delta))$;
- $\ell_0 = \Theta(\delta^2 n/\log(1/\delta))$; $\ell_i = \ell_{i-1}/(1+\delta'/4)$ for each $0 < i < s$, whereby $s = O(\log(\widetilde{N})\log(1/\delta)/\delta)$; therefore, $\ell_s = \log(1/\varepsilon)\log(1/\delta)/\delta$;
- $k = O(\log(1/\varepsilon))$.

The following is a description of the extractor E:

1. **Input:** $x_1 \in \{0,1\}^{\widetilde{N}}$, $y_1, \ldots, y_s \in \{0,1\}^{k \log \widetilde{N}}$, $y_0 \in \{0,1\}^{2\ell_s}$.
2. **Output:** $C(D(x, y_1, \ldots, y_s), y_0)$. (Here, D and C are used with the parameters specified above.)

It was shown in [84] that if x is from a δ-source and the y's are uniform, then the output of the function E is ε-close to a uniform m-bit string, where $m = \Omega(\delta^2 n/\log(1/\delta))$.

Claim 4.18. The function E can be computed using a circuit of size $\widetilde{N} \cdot \operatorname{polylog}\left(\widetilde{N}/\varepsilon\right)$.

Proof. This follows from Claim 4.16 and Claim 4.17. □

4.2.2 Local PRG that Fools Subcubic De Morgan Formulas

We need the following pseudorandom shrinkage lemma for De Morgan formulas, which says that there exists a p-regular restriction, where the unrestricted variables are selected pseudorandomly and the restricted variables are fixed truly randomly, such that *with high probability* the size of the restricted formula will "shrink" by a factor of p^2.

Lemma 4.19 (Pseudorandom shrinkage lemma, [57, Lemma 4.8][1])**.** *There exists a constant $c_0 > 0$ such that the following holds. For any constant $c > c_0$, any $s \geq N$, $p \geq s^{-1/2}$, and any De Morgan formula F on N variables of size s, there exists a p-regular pseudorandom selection $\mathcal{D}$ over N variables that is samplable using $r = 2^{O(\log^{2/3} s)}$ random bits such that*

$$\Pr_{\sigma \sim \mathcal{D}, x \sim \{0,1\}^N} \left[L\big(F|_{(\sigma,x)}\big) \geq 2^{3 \cdot c \cdot \log^{2/3} s} \cdot p^2 \cdot s \right] \leq s^{-c}.$$

Moreover, there exists a circuit of size $2^{O(\log^{2/3} s)}$ such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th coordinate of $\mathcal{D}(z)$.

We are now ready to show our PRG in Lemma 4.7.

Proof of Lemma 4.7. The construction is as follows: We first sample a p-regular pseudorandom selection from Lemma 4.19. Then, we fill the star coordinates, specified by the pseudorandom

[1] The pseudorandom shrinkage lemma in [57] is not stated in this form, but rather selects the unrestricted variables and fixes the restricted variables both pseudorandomly (based on limited independence). This immediately implies the above lemma, where the restricted variables are set independently (and hence also k-wise independently, for any k). Further, the last statement follows from the fact that the restricted variables are chosen by a k-wise independent distribution, which can be computed locally; see Lemma 2.8.

selection, in the output string with the output of some extractor which takes a min-entropy source sample and a short seed. (More precisely, the star coordinates are filled with the output of some limited-independence generator that takes the output of an extractor as a seed.) We then sample another pseudorandom selection, and fill the star coordinates specified by this pseudorandom selection but this time only for those that have not been filled in previous steps, again with the output of the same extractor using the *same min-entropy source sample* but a *different short seed*. We continue this way until all the coordinates are filled.

More formally, our PRG uses the following parameters:[2]

- $p = 1/s^{1/3}$, the expected fraction of unrestricted variables in each of the pseudorandom selections;
- $\varepsilon = 1/\mathrm{poly}(N)$ and $\varepsilon_0 = \varepsilon/(10t)$, which specify the error of the PRG;
- $t = \ln(4N/\varepsilon)/p = s^{1/3} \cdot O(\log N)$, the number of steps needed so that all the coordinates will be filled with probability $1 - \varepsilon/4$;
- $s_0 = p^2 \cdot s \cdot 2^{O\left(\log^{2/3} s\right)} = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, the size of the formula after being simplified by a pseudorandom restriction;
- $k \geq s_0 = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, the amount of independence needed to fool the simplified formula, and $r_k = k \cdot \log N$ the seed length for the k-wise independent generator;
- $\widetilde{N}$, the length of the min-entropy source for the extractor, which is such that $\widetilde{N} \geq 2 \cdot \log(1/\varepsilon_0) + c \cdot s_0 \cdot \log s_0$, where $c > 0$ is some constant, and that $\Omega(\widetilde{N}) \geq r_k$. We can take $\widetilde{N} = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$;

[2] In fact, there are mainly two types of parameters here. Those that are close to $s^{1/3}$, which are $1/p, t, s_0, k, N$, and those that are close to $N^{o(1)}$, which are d and ℓ.

- $d = \text{polylog}(\widetilde{N}/\varepsilon_0) = \text{polylog}(N)$, the seed length of the extractor;
- $\ell = 2^{O(\log^{2/3} s)}$, the number of random bits for sampling a pseudorandom selection.

Construction. The PRG takes a seed $(X, Y_1, \ldots, Y_t, \gamma_1, \ldots, \gamma_t) \in \{0,1\}^r$, where

- $X \in \{0,1\}^{\widetilde{N}}$ is the min-entropy source sample of an extractor,
- $Y_i \in \{0,1\}^{\text{polylog}(N)}$, for each $i \in [t]$, is the seed of an extractor, and
- $\gamma_i \in \{0,1\}^{\ell}$, for each $i \in [t]$, is the seed for sampling a pseudorandom selection.

The construction of the PRG proceeds in the following two stages.

Stage 1. Compute a sequence of t p-regular pseudorandom selections

$$\sigma_1, \ldots, \sigma_t,$$

using Lemma 4.19, with the seeds $\gamma_1, \ldots, \gamma_t$. Below, we denote the star coordinates in σ_i by $\sigma_i^{-1}(*)$. Let $S_1, \ldots, S_t \subseteq [N]$ be t disjoint sets defined by

$$S_i = \sigma_i^{-1}(*) \setminus (S_1 \cup \cdots \cup S_{i-1}).$$

Stage 2. Define $Z_1, \ldots, Z_t \in \{0,1\}^N$ by

$$Z_i = G_k(E(X, Y_i)),$$

where $E : \{0,1\}^{\widetilde{N}} \times \{0,1\}^d \to \{0,1\}^{\Omega(\widetilde{N})}$ is an $(\widetilde{N}/2, \varepsilon_0)$-extractor and $G_k : \{0,1\}^{r_k} \to \{0,1\}^N$ is a k-wise independent generator. The final output of our PRG is the binary string that has the values $Z_i|_{S_i}$ in the positions indexed by S_i, for all $i \in [t]$, where $Z_i|_{S_i}$ denotes the bit values of Z_i projected to the set S_i. (We fix those positions that are not in any of the S_i's to be 0.) Stage 2 of the PRG construction is depicted in Figure 4.1.

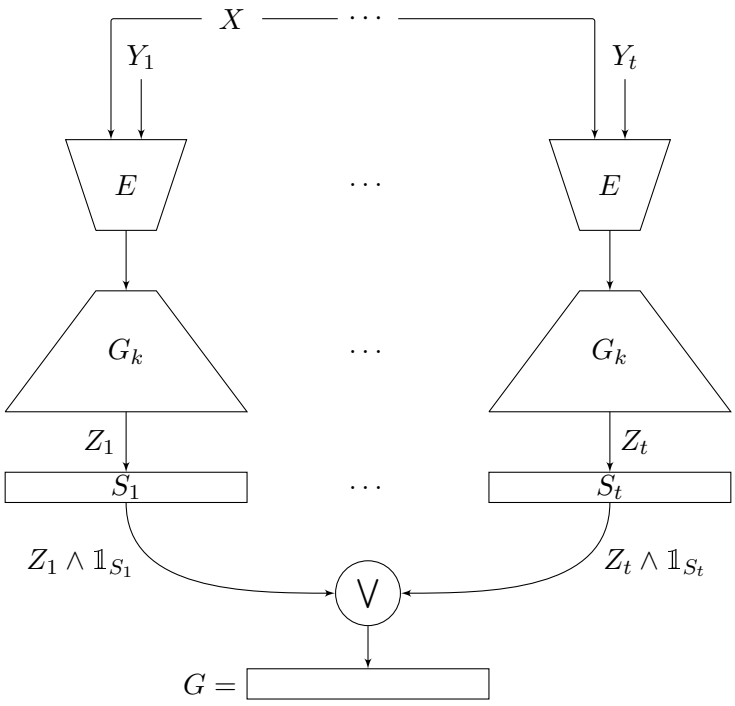

Figure 4.1: Construction of the PRG in Lemma 4.7, Stage 2. For each $i \in [t]$, $\mathbb{1}_{S_i} \in \{0,1\}^N$ denotes the characteristic Boolean vector of the set S_i, where $S_i \subseteq [N]$ is the set of star coordinates, in the i-th pseudorandom selection, that did not appear in the preceding sets $S_1, \ldots, S_{i-1}$. Also, $\wedge$ denotes a coordinate-wise AND operation (i.e., coordinate-wise *multiplication* of Boolean vectors) and $\bigvee$ is a coordinate-wise OR operation.

Correctness. Next, we show that the above PRG ε-fools N-variate formulas of size s. First, note that, by our choice of t, with probability $1-\varepsilon/4$, $S = S_1 \cup \cdots \cup S_t$ covers all N coordinates. To proceed, we will use a *hybrid argument*. Let G denote the distribution given by the PRG described above. Let U be the uniform distribution. Note that if in the above construction we replace Z_i, for all $i \in [t]$, with U, then we would get a uniform distribution. Now we can start from there and we will gradually replace U with the Z_i's step-by-step for a total of t steps. We will argue that after each replacement step, the expected value of the function does not change

by much. Let B_i be the distribution where we have replaced U with Z_i in the first i steps and $S = [N]$. That is,

$$B_i = \left(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, U|_{S_{i+1}}, \ldots, U|_{S_t} \right) = \left(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, U|_{\ S_{i+1} \cup \cdots \cup S_t} \right)$$

and we want to show that $|\mathbf{E}[f(U)] - \mathbf{E}[f(G)]| = |\mathbf{E}[f(B_0)] - \mathbf{E}[f(B_t)]| \leq \varepsilon$. Let

$$\begin{aligned} A_i &= \left(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, U|_{S_{i+1}}, \ldots, U|_{S_t}, U|_{[N] \setminus S} \right) \\ &= \left(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, U|_{\ S_{i+1} \cup \cdots \cup S_t \cup ([N] \setminus S)} \right) \end{aligned}$$

be the version of the distribution B_i in the case where $S \subsetneq [N]$. Let $\mathcal{C}$ denote the event $S = [N]$; by Lemma 4.3, for all i, we get that

$$|\mathbf{E}[f(A_i)] - \mathbf{E}[f(B_i)]| \leq |\mathbf{E}[f(A_i) \mid \mathcal{C}] - \mathbf{E}[f(B_i) \mid \mathcal{C}]| + \Pr[\neg \mathcal{C}] \leq 0 + \varepsilon/4 = \varepsilon/4.$$

Therefore, it would suffice to establish $|\mathbf{E}[f(A_0)] - \mathbf{E}[f(A_t)]| \leq \varepsilon/2$ since this inequality would imply the desired $|\mathbf{E}[f(B_0)] - \mathbf{E}[f(B_t)]| \leq \varepsilon$.

Note that using the distributions would B_i require that $S = [N]$ and this could result in dependencies among the sets S_i. This is the reason for introducing the distributions A_i; we shall later make use of the fact that the selections σ_i, that come up in the definitions of the sets S_i, are independent.

Now, for the sake of contradiction, suppose there exists a size-s formula f on N variables such that

$$|\mathbf{E}[f(A_0)] - \mathbf{E}[f(A_t)]| > \varepsilon/2.$$

By the triangle inequality, there exists an $0 \leq i < t$ such that

$$|\mathbf{E}[f(A_i)] - \mathbf{E}[f(A_{i+1})]| > \varepsilon/(2t). \tag{4.1}$$

Let us say that both the expectations in Equation (4.1) are over

$$\sigma_1, \ldots, \sigma_{i+1}, Y_1, \ldots, Y_{i+1}, X, U,$$

and we remove the absolute value without loss of generality. Then, we have

$$\mathop{\mathbf{E}}_{\substack{\sigma_1,\ldots,\sigma_i,\\ Y_1,\ldots,Y_i,\\ X}} \left[\mathop{\mathbf{E}}_{\sigma_{i+1},Y_{i+1},U} [f(A_i)] - \mathop{\mathbf{E}}_{\sigma_{i+1},Y_{i+1},U} [f(A_{i+1})] \right] > \varepsilon/(2t). \tag{4.2}$$

Denote $W_i = (\sigma_1, \ldots, \sigma_i, Y_1, \ldots, Y_i, X)$, and let f' be the random function (where the randomness is over W_i) defined as

$$f' = f(Z_1|_{S_1}, \ldots, Z_i|_{S_i}, \quad \cdot \quad).$$

That is, f' is the restricted function after the first i steps. Then, the LHS of Equation (4.2) becomes

$$\begin{aligned} \mathop{\mathbf{E}}_{W_i} \big[\mathop{\mathbf{E}}_{\sigma_{i+1},U} \big[f'(U|_{S_{i+1}}, U|_{S_{i+2}\cup\cdots\cup S_t\cup([N]\setminus S)}) \big] \\ - \mathop{\mathbf{E}}_{\sigma_{i+1},Y_{i+1},U} \big[f'(Z_{i+1}|_{S_{i+1}}, U|_{S_{i+2}\cup\cdots\cup S_t\cup([N]\setminus S)}) \big] \big]. \end{aligned} \tag{4.3}$$

Note that, at this point, we can view $\rho_{i+1} = (\sigma_{i+1}, U)$ as a pseudorandom restriction (in the sense of Lemma 4.19) applied to f'. Next, let f'' be the random function defined as the restricted function of f' under ρ_{i+1} (note that the randomness is over W_i, and also the pseudorandom restriction ρ_{i+1}). Now Equation (4.3) becomes

$$\mathop{\mathbf{E}}_{W_i,\rho_{i+1}} \left[\mathop{\mathbf{E}}_{U} \big[f''(U) \big] - \mathop{\mathbf{E}}_{Y_{i+1}} \big[f''(Z_{i+1}) \big] \right]. \tag{4.4}$$

Note that in the above, we abuse the notation and use U and Z_{i+1} to denote $U|_{S_{i+1}}$ and $Z_{i+1}|_{S_{i+1}}$, respectively.

Next, we want to show that the difference between the two expectations in Equation (4.4) is at most $3\varepsilon_0 = 3\varepsilon/(10t) \leq \varepsilon/(2t)$, which would give a contradiction, by Equation (4.2). The

intuition is the following. On the one hand, f'' is obtained by a pseudorandom restriction ρ_{i+1}, and so, with high probability, it has size at most s_0. On the other hand, Z_{i+1} is obtained using an extractor that is supposed to extract enough random bits for an s_0-independent generator.

The issue, however, is that f'' depends on X, the source sample of the extractor. Therefore, f'' may contain information about X, so that X is not truly random anymore. Nonetheless, being a formula of size at most s_0, f'' cannot contain too much information, and so cannot take too much entropy away from X. We make this argument more formal next.

Let us define the set of good functions for f'', namely

$$\mathcal{F} = \left\{ g \mid L(g) \leq s_0 \quad \text{and} \quad \mathop{\mathbf{Pr}}_{W_i, \rho_{i+1}} [f'' = g] \geq \varepsilon_0 / s_0^{cs_0} \right\},$$

where c is some constant. Let $\mathcal{E}$ denote the event $f'' \in \mathcal{F}$. We first show the following.

Claim 4.20. It is the case that $\mathbf{Pr}[\neg\mathcal{E}] \leq 2\varepsilon_0$.

Proof of Claim 4.20. We have

$$\begin{aligned}\mathbf{Pr}[\neg\mathcal{E}] &= \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') > s_0)] + \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') \leq s_0)] \\ &\leq \mathbf{Pr}[(L(f'') > s_0)] + \mathbf{Pr}[(f'' \notin \mathcal{F}) \wedge (L(f'') \leq s_0)].\end{aligned}$$

Note that, by the pseudorandom shrinkage lemma (Lemma 4.19), we have

$$\mathbf{Pr}[L(f'') > s_0] \leq \varepsilon_0;$$

in fact, our choices of s_0 and ε_0 were informed by our intention to make the above inequality hold. Also note that under the condition that $L(f'') \leq s_0$, there can be at most $s_0^{O(s_0)}$ choices for f'', since a formula of size s_0 can be specified using $O(s_0 \log s_0)$ bits (Lemma 2.1). Therefore,

$$\mathbf{Pr}\big[(f'' \notin \mathcal{F}) \wedge (L(f'') \leq s_0)\big] \leq s_0^{O(s_0)} \cdot \varepsilon_0 / s_0^{cs_0} \leq \varepsilon_0. \qquad \square$$

Let us now analyze Equation (4.4) while conditioning on the event $\mathcal{E}$. We show the following.

Claim 4.21. It is the case that $\mathbf{E}\left[f''(U) \mid \mathcal{E}\right] - \mathbf{E}\left[f''(Z_{i+1}) \mid \mathcal{E}\right] \leq \varepsilon_0$.

Proof of Claim 4.21. First note that conditioning on $\mathcal{E}$, X still has a large min-entropy. More precisely, for every $g \in \mathcal{F}$ it is the case that

$$H_\infty\big(X \mid f'' = g\big) \geq \widetilde{N}/2.$$

This is because, for every x, we have

$$\mathbf{Pr}[X = x \mid f'' = g] \leq \frac{\mathbf{Pr}[X = x]}{\mathbf{Pr}[f'' = g]} \leq \frac{2^{-\widetilde{N}}}{\varepsilon_0 / s_0^{c \cdot s_0}} = 2^{-(\widetilde{N} - \log(1/\varepsilon_0) - c \cdot s_0 \cdot \log s_0)} \leq 2^{-\widetilde{N}/2}.$$

Then, by the definition of the extractor, we have

$$\mathbf{E}\left[f''(G_k(U)) \mid \mathcal{E}\right] - \mathbf{E}\left[f''(Z_{i+1}) \mid \mathcal{E}\right] \leq \varepsilon_0.$$

Finally, note that

$$\mathbf{E}\left[f''(G_k(U)) \mid \mathcal{E}\right] = \mathbf{E}\left[f''(U) \mid \mathcal{E}\right],$$

since s_0-wise independent distributions fool size-s_0 formulas. □

Combining Claim 4.20, Claim 4.21, and Lemma 4.3, we get that the quantity in Equation (4.4) is at most $3\varepsilon_0$, which leads to a contradiction. This completes the proof of correctness.

Locality. To see that the j-th bit of the PRG can be computed using a circuit of size $s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, we observe the following equivalent construction:

1. Compute the j-th bits of the t pseudorandom selections $(\sigma_1)_j, \ldots, (\sigma_t)_j$.
2. Retrieve Y_q, where q is the smallest integer such that $(\sigma_q)_j$ is a star.
3. Compute $(Z_q)_j = G_k(E(X, Y_q))_j$ as the j-th bit of the PRG.

Note that Step 1 can be done using a circuit of size $t \cdot 2^{O\left(\log^{2/3} s\right)} = s^{1/3} \cdot 2^{O\left(\log^{2/3} s\right)}$, by the pseudorandom shrinkage lemma (Lemma 4.19). Also, Step 2 can be done by first computing q from the sequence $((\sigma_i)_j)_{i\in[t]}$ using a circuit of size $\widetilde{O}(t)$ (Lemma 4.4), and then outputting Y_q from $(Y_i)_{i\in[t]}$ using a circuit of size $t \cdot \mathrm{polylog}(N)$ (Lemma 4.5). Finally, Step 3 can be done by a circuit of size $\widetilde{O}(\widetilde{N})$ using the efficient extractor (Lemma 4.11) and the limited-independence generator (Lemma 2.8). □

4.3 MCSP Lower Bounds Against Arbitrary Basis Formulas and Branching Programs

Here, we prove MCSP lower bounds against formulas, over an arbitrary basis, and branching programs. These lower bounds are obtained similarly to those for De Morgan formulas in the previous section. The idea is to construct local PRGs against these models by modifying the PRGs in [57].

The following pseudorandom shrinkage lemma for formulas over an arbitrary basis as well as branching programs is an analogue of Lemma 4.19.

Lemma 4.22 ([57, Lemma 4.2 and Lemma 5.3])**.** *There exists a constant $c_0 > 0$ such that the following holds. For any constant $c > c_0$ and any $s \geq N$, let $p = s^{-1/2}$ and F be a formula over any basis (or a branching program) on N variables of size s; then, there exists a p-regular pseudorandom selection $\mathcal{D}$ over N variables that is samplable using $r = \mathrm{polylog}(N)$ random bits such that*

$$\Pr_{\sigma\sim\mathcal{D}, x\sim\{0,1\}^N}\left[L\left(F|_{(\sigma,x)}\right) \geq 2^{3\cdot\sqrt{c\cdot\log s}} \cdot p \cdot s\right] \leq 2 \cdot s^{-c}.$$

Moreover, there exists a circuit of size $2^{O(\log^{2/3} s)}$ such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th bit of $\mathcal{D}(z)$.

Using the above pseudorandom shrinkage lemma and an argument as in the proof of the local PRG that fools De Morgan formulas (Lemma 4.7), we get the following local PRGs.

Lemma 4.23. *For any $s \geq N$, there exists a $s^{1/2} \cdot 2^{O(\sqrt{\log s})}$-local PRG of seed length $s^{1/2} \cdot 2^{O(\sqrt{\log s})}$ that $(1/3)$-fools size-s formulas over an arbitrary basis (or branching programs) that have N inputs.*

The MCSP lower bound in Theorem 1.2 follows from Lemma 4.23 and Theorem 3.4.

Notes

The results of this chapter are products of joint work with Mahdi Cheraghchi, Valentine Kabanets, and Zhenjian Lu [29].

Chapter 5

MCSP Lower Bounds Against Constant Depth Circuits

In 2006, Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [6] proved that MCSP cannot be computed by polynomial-size AC^0 circuits. In this chapter, we improve the superpolynomial lower bound of Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger by showing that computing MCSP on input a truth table of length N requires $2^{\Omega\left(N^{1/(d+1.01)}\right)}$-size depth-$d$ AC^0 circuits. Our AC^0 lower bound for MCSP matches the best-known AC^0 lower bound (for Parity) up to a small *additive* constant in the depth.

To prove our lower bound, we use a local PRG that fools AC^0. To get a lower bound matching the one in Theorem 1.3, we can use the state-of-the-art PRG that fools AC^0 by Trevisan and Xue [109], which has a seed length of $(\log s)^{d+O(1)}$ for size-s depth-d AC^0 circuits. By a careful analysis of the construction of this PRG, we can show that the Trevisan-Xue PRG is local and can be used to get an MCSP lower bound that is close to the one stated in Theorem 1.3. However, in this chapter, we will present a more direct proof of such a lower

bound by using the pseudorandom switching lemma for constant-depth circuits, which is due to Trevisan and Xue [109] as well, and is a key ingredient in their PRG.

The idea is to show that for any small-depth circuit of size less than the claimed lower bound, there is some locally computable restriction that turns the circuit into a constant function, but leaves many variables unrestricted. However, MCSP cannot be constant under such a restriction, because depending on the partial assignment to the unrestricted variables, the resulting input function (which is composed of the restriction and the partial assignment) can be either easy or hard. Such an approach based on pseudorandom restrictions can also be applied to the special case of depth-2 circuits to get optimal CNF (and DNF) lower bounds for MCSP.

Moreover, for the special case of depth-2 circuits (i.e., CNFs or DNFs), we get an optimal lower bound of $2^{\Omega(N)}$ for MCSP.

The improved lower bounds for MCSP (Theorem 1.3 and Theorem 1.4) are proved in Section 5.1.

5.1 Improved AC^0 Lower Bounds for MCSP

In this section we show lower bounds for MCSP against constant-depth circuits, that improve the state-of-the-art lower bounds (by Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [6]) for MCSP against this kind of circuits.

5.1.1 The Case of Depth $d > 2$

We first show an improved lower bound against depth-d circuits that almost matches the $2^{\Omega\left(N^{1/(d-1)}\right)}$ lower bound for Parity.

Theorem 5.1 (Theorem 1.3, restated)**.** *For every $d > 2$ and every constant $\gamma > 0$, any depth-d AC^0 circuit computing* MCSP *on truth tables of length N must have size $2^{\Omega\left(N^{1/(d+1+\gamma)}\right)}$.*

The above result is proved using the following structural property of small-depth circuits, which says that, for any such circuit, there exists some locally computable restriction that simplifies the circuits to a constant while leaving many variables unrestricted.

Lemma 5.2. *For any size-s depth-d circuit C, there exists a restriction $\rho \in \{0,1,*\}^N$ such that*

- $C|_\rho$ *is a constant function,*
- $\left|\rho^{-1}(*)\right| \geq \frac{N}{O(\log s)^{d-2}} - \log s$*, and*
- *there exists a circuit of size $d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$ such that, given $j \in \{0,1\}^{\log N}$, the circuit computes the j-th coordinate of ρ.*

We now prove Theorem 5.1 using Lemma 5.2.

Proof of Theorem 5.1. Let C be a depth-d AC^0 circuit on $\{0,1\}^N \times \{0,1\}^{\log N}$ such that C computes MCSP on truth tables of length N, and let s be the size of C.

For a size parameter $\lambda = d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$, let $C' = C(\cdot, \lambda)$. Let ρ be a restriction from Lemma 5.2 for C'. By Lemma 5.2, we have that $C'|_\rho$ is a constant function. First, note that

$$C'\big|_\rho\left(0^{|\rho^{-1}(*)|}\right) = 1.$$

To see this, note that

$$C'\big|_\rho\left(0^{|\rho^{-1}(*)|}\right) = C\left(\mathtt{tt}(f), \lambda\right),$$

where C computes MCSP and $f : \{0,1\}^{\log N} \to \{0,1\}$ is the following:

$$f(j) = \begin{cases} 0 & \text{if } \rho_j = 0 \text{ or } \rho_j = *, \\ 1 & \text{if } \rho_j = 1. \end{cases}$$

By Item 3 of Lemma 5.2, such a function f can be computed by a λ-size circuit. On the other hand, there can be $2^{|\rho^{-1}(*)|}$ different functions corresponding to the different partial assignments to the unrestricted variables. Since there are at most $2^{O(\lambda \log \lambda)}$ different circuits of size at most λ, in order for $C'|_\rho$ to be constant and equal to 1, we must have

$$2^{O(\lambda \log \lambda)} \geq 2^{|\rho^{-1}(*)|} = 2^{\frac{N}{O(\log s)^{d-2}} - \log s},$$

which, by a simple calculation, implies $s = 2^{\Omega\left(N^{1/(d+1+\gamma)}\right)}$, for any constant $\gamma > 0$. □

The proof of Lemma 5.2 uses the pseudorandom switching lemma due to Trevisan and Xue [109], which we revisit below. The (pseudorandom) switching lemma says that a depth-2 circuit is likely to be simplified after being hit by a pseudorandom restriction.

Below, when we refer to the size of a DNF or CNF we mean the number of its terms or clauses, respectively.

Lemma 5.3 (Pseudorandom switching lemma, [109, Lemma 7])**.** *For any integers $t, w > 0$, $s \geq N$, and any $0 < p, \varepsilon_0 < 1$, let F be an N-variate w-CNF or w-DNF of size s, and let $\mathcal{D}$ be a distribution of p-regular pseudorandom restrictions over $\{0,1\}^{N \cdot \log(1/p)} \times \{0,1\}^N$ that ε_0-fools $\left(s_0 = s \cdot 2^{w \cdot (\log(1/p)+1)}\right)$-clause CNFs, then*

$$\Pr_{\rho \sim \mathcal{D}}[F|_\rho \text{ does not have a depth-}t \text{ decision tree}] \leq 2^{t+w+1} \cdot (5pw)^w + \varepsilon_0 \cdot 2^{(t+1)(2w+\log s)}.$$

Lemma 5.4 (Following [109, Theorem 11])**.** *For any integers $d, t > 0$, $s \geq N$, and any $(480/N)^{1/(d-2)} < p < 1$ and $0 < \varepsilon_0 < 1$, there exists a distribution $\mathcal{D}$ over $\{0,1\}^{N \cdot \log(1/p)} \times \{0,1\}^N$ for sampling a pseudorandom restriction such that*

- *for any size-s depth-d circuit C on N variables, we have that*

$$\Pr_{\rho\sim\mathcal{D}}\left[C|_\rho \text{ is not a } t\text{-DNF or } t\text{-CNF}\right] \leq s\cdot\left(2^{2t+1}\cdot(10pt)^t+\varepsilon_0\cdot 2^{(t+1)(4t+\log s)}\right),$$

- *with probability at least $2/3$ the number of unrestricted variables is $\frac{p^{d-2}}{80}\cdot N$, and*

- *there exists a circuit of size $d\cdot k\cdot\widetilde{O}(\log N)$ such that, given $j\in\{0,1\}^{\log N}$ and a seed $z\in\{0,1\}^{d\cdot k\cdot O(\log N)}$, the circuit computes the j-th coordinate of $\rho=\mathcal{D}(z)$ (as an element in $\{0,1,*\}$), where*

$$k=O\big((\log s+t\cdot\log(1/p))^2+(\log s+t\cdot\log(1/p))\cdot\log(1/\varepsilon_0)\big).$$

Proof. The proof is similar to that of Theorem 11 in [109]. Let $x_1,\ldots,x_N$ be the input variables. We fist add at the bottom of C a dummy layer of N OR or AND gates (depending on whether the first layer of C consists of AND or OR gates, respectively) that have fan-in 2, and are of the form $0\vee x_i=x_i$ or $1\wedge x_i=x_i$ for all $i\in[N]$. This layer just propagates the input variables, and does nothing else; we add it for technical reasons only, to facilitate the application of Lemma 5.3.

The idea now is to apply the pseudorandom switching lemma (Lemma 5.3) repeatedly. Each time, we sample a pseudorandom restriction using some distribution that ε_0-fools CNFs of size $s_0=s\cdot 2^{w\cdot(\log(1/p)+1)}$, for $w=t$ (apart from the first restriction, where $w=2$). By Lemma 5.3, each time, with high probability, the two bottom layers can be computed by depth-t decision trees, so we can switch them to t-DNFs or t-CNFs, and hence reduce the depth of the circuit by one as we merge them with the layer above.

One difference here from the argument in [109] is that we only apply the pseudorandom switching lemma $d-1$ times, instead of d times, since we only need the final restricted circuit to be a t-DNF or t-CNF (rather than a depth-t decision tree as in the original statement of [109],

which requires an additional application of the pseudorandom switching lemma). Note that we use parameter $p_1 = 1/2^6$ for the first iteration. Another difference is that, to sample a pseudorandom restriction, we use a k-wise independent distribution (say over $[1/p]^{2N}$), instead of using the PRG that fools depth-2 circuits in [32], where

$$k = O(\log(s_0/\varepsilon_0) \cdot \log s_0) = O\big((\log(s) + t \cdot \log(1/p))^2 + (\log(s) + t \cdot \log(1/p)) \cdot \log(1/\varepsilon_0)\big)\,,$$

and we use the fact that such a k-wise independent distribution ε_0-fools s_0-clause CNFs [107, Theorem 22].[1]

According to the discussion above, we have

$$\begin{aligned}\mathop{\mathbf{Pr}}_{\rho\sim\mathcal{D}}\Big[C|_\rho \text{ is not a } t\text{-}\mathsf{DNF} \text{ or } t\text{-}\mathsf{CNF}\Big] &\le 2^{t+3}\cdot(10p)^2 + \varepsilon_0 \cdot 2^{(t+1)(4+\log s)} \\ &\quad + (d-2)\left(2^{2t+1}\cdot(5pt)^t + \varepsilon_0\cdot 2^{(t+1)(2t+\log s)}\right) \\ &\le s\cdot\left(2^{2t+1}\cdot(10pt)^t + \varepsilon_0\cdot 2^{(t+1)(4t+\log s)}\right)\end{aligned}$$

by a union bound and Lemma 5.3, as $C|_\rho$ is not a t-DNF or t-CNF only if some of the $d-1$ applications of the pseudorandom switching lemma (Lemma 5.3) fails. The desired inequality of Item 1 now follows from the fact that $d - 1 \le s$, as the depth d of a circuit is at most $s+1$ for s its size.

The expected number of unrestricted variables is $p_1 \cdot p^{d-2} \cdot N = \frac{p^{d-2}}{40} \cdot N$. Then Item 2 follows from the fact that the random restriction is pairwise independent and Proposition 4.2. Indeed, the probability that the number of unrestricted variables is at least $\frac{p^{d-2}}{80} \cdot N$ is

$$1 - \mathbf{Pr}\left[X \le \frac{p^{d-2}}{40}\cdot N/2\right] \ge 1 - \frac{4}{\frac{p^{d-2}}{40}\cdot N}$$

[1] The PRG in [32] is based on a *small-biased distribution*. While it has smaller seed length, compared to a k-wise independent distribution, it does not seem to offer any advantage in terms of the local circuit complexity of computing the PRG.

$$
\begin{aligned}
&= 1 - \frac{160}{p^{d-2} \cdot N} \\
&\geq 1 - \frac{160}{\left((480/N)^{1/(d-2)}\right)^{d-2} \cdot N} \\
&= 1 - \frac{1}{3} = \frac{2}{3}.
\end{aligned}
$$

Finally, one may get Item 3 using Lemma 2.8. That is, by Lemma 2.8, we get that there exists a circuit of size

$$k \cdot \max\{\widetilde{O}(\log N), \widetilde{O}(\log(1/p))\} \leq k \cdot \widetilde{O}(\log N) \qquad (\text{since } 1/p > (N/480)^{1/(d-2)})$$

such that, given $j \in \{0,1\}^{\log N}$ and a seed $z \in \{0,1\}^{d \cdot k \cdot O(\log N)}$, the circuit computes the j-th coordinate of a restriction $\rho = \mathcal{D}'(z)$, where $\mathcal{D}'$ is a k-wise independent distribution over $\{0,1\}^{N \cdot \log(1/p)} \times \{0,1\}^N$. The desired upper bound of Item 3 now follows from the fact that we apply $d-1$ pseudorandom restrictions, in total, and their composition gives rise to the distribution $\mathcal{D}$ that appears in the statement of Lemma 5.4. □

We are now ready to show Lemma 5.2.

Proof of Lemma 5.2. By Lemma 5.4, using the parameters $t = O(\log s)$, $p = 1/O(\log s)$, and $\varepsilon_0 = 1/2^{O(\log^2 s)}$, we get a restriction ρ_0 such that the circuit restricted by ρ_0 is a width-$O(\log s)$ DNF or CNF, with probability at least $1 - 1/\text{poly}(N)$.

By Item 2 of Lemma 5.4, ρ_0 leaves at least $\frac{N}{O(\log s)^{d-2}}$ variables unrestricted, with constant probability. Therefore, by a union bound, with some constant probability, we get a restriction ρ_0 that both simplifies the circuit to be a width-$(\log s)$ DNF or CNF and that leaves $\frac{N}{O(\log s)^{d-2}}$ variables unrestricted. Note that once we have such a restriction, we can make the restricted circuit constant by further fixing at most $\log s$ variables; denote this restriction by ρ_1. The final restriction is $\rho = \rho_0 \circ \rho_1$. This concludes the proofs of Item 1 and Item 2.

We now show the last item. Note that our final restriction consists of two parts, ρ_0 and ρ_1, where ρ_0 is a restriction from Lemma 5.4 and ρ_1 is a restriction that fixes $\log s$ variables. To compute the final restriction, given an index $j \in \{0,1\}^{\log N}$, we can first check if the j-th variable is fixed by ρ_1 and output the fixing value if it is the case. This can be done by hard-wiring the $\log s$ variables that are fixed by ρ_1 and their corresponding fixing values. It is straightforward to see that the above can be done using a circuit of size at most $O(\log s \cdot \log N)$. Otherwise, we can output the j-th coordinate of ρ_0, which can be done with a circuit of size $d \cdot \log(N) \cdot \widetilde{O}\left(\log^3 s\right)$, by Item 3 of Lemma 5.4. □

5.1.2 The Case of Depth 2

Here, we show that computing MCSP requires depth-2 circuits of almost maximum size.

Theorem 5.5 (Theorem 1.4, restated)**.** *Any* CNF *or* DNF *computing* MCSP *on truth tables of length N must have size $2^{\Omega(N)}$.*

To prove Theorem 5.5, we will utilize the following lemma and corollary.

Lemma 5.6. *Let $0 < \delta < 1$. Any N-variate* CNF *or* DNF *of size $s \leq 2^{\delta N}$ can be fixed to a constant by applying a restriction that sets $O\left(\sqrt{\delta}N\right)$ variables.*

Proof. We will show the lemma for the case of DNFs. The proof can be easily adapted to the case of CNFs. Fix a constant $A := 1/\sqrt{\delta}$. We choose the restriction in question in two phases. In Phase 1, we show that we can set at most $O\left(\sqrt{\delta}N\right)$ variables to get the width of the DNF down to $A \log s = \sqrt{\delta}N$. In Phase 2, we can easily set any term of the remaining DNF to fix the function. Since Phase 2 is trivial, let us henceforth focus on Phase 1.

To this end, imagine that we choose a uniformly random input variable x_i (for some i) and set it to a random value. If T is any term in the DNF of size greater than $A \log s$, then T is

set to 0 with probability at least

$$\frac{A \log s}{2N}.$$

Repeating this process $t := 2\sqrt{\delta}N$ times, we see that the probability that T survives is at most

$$\left(1 - \frac{A \log s}{2N}\right)^t = \left(1 - \frac{\log s}{2\sqrt{\delta}N}\right)^{2\sqrt{\delta}N} \leq \exp(-\log s) < \frac{1}{s}.$$

By a union bound over the (at most) s terms of the DNF in question, there is a restriction ρ that restricts $O\left(\sqrt{\delta}N\right)$ variables such that ρ sets all the terms of width greater than $A \log s$ to 0. This completes Phase 1. □

Corollary 5.7. *For any size-s depth-2 circuit C, there exists a restriction $\rho \in \{0,1,*\}^N$ such that*

- $C|_\rho$ *is a constant function,*
- $\left|\rho^{-1}(*)\right| \geq \Omega(N)$*, and*
- *there exists a circuit of size* $\log s / \Omega(\log \log s)$ *such that, given $j \in \{0,1\}^{\log N}$, the circuit computes the j-th coordinate of ρ.*

Proof. By Lemma 5.6. We shall verify that all of three properties of Corollary 5.7 hold for the restriction ρ that is proved to exist by Lemma 5.6. Items 1 and 2 trivially hold, as $C|_\rho$ is a constant by the construction of ρ and $\left|\rho^{-1}(\{0,1\})\right| = O\left(\sqrt{\delta}N\right)$, respectively. Item 3 follows by a result of Lupanov [40, Theorem 2.2] for the (biased) Boolean function that sets all the unrestricted variables to 0. □

One may now prove Theorem 5.5 by using Corollary 5.7, in a way *identical* to how Theorem 5.1 is proved by using Lemma 5.2.

Notes

The results of this chapter are products of joint work with Mahdi Cheraghchi, Valentine Kabanets, and Zhenjian Lu [29].

Chapter 6

MCSP Lower Bounds Against Augmented Constant Depth Circuits

It is well known in the field of derandomization that, if we have a function that is average-case hard against some circuit class $\mathfrak{C}$, we can get a PRG by plugging the hard function into the Nisan-Wigderson framework [83] (provided that the hard function is not too hard to compute and that $\mathfrak{C}$ satisfies some mild conditions).

The construction involves computing some combinatorial design with some suitably chosen parameters; a design is a list of subsets (over some universe) that have some combinatorial properties (see Definition 6.1). Also, to compute a single bit of such PRG, we need to compute the corresponding subset of the design. There are known design constructions such that any single subset of the design can be computed efficiently and locally (without computing the whole design). The circuit complexity of computing a subset, which is related to the local complexity of the Nisan-Wigderson PRG, will depend on the parameters that we choose for the design, which in turn will depend on the "usefulness" of the average-case hard function.

Therefore, using such a local design, we can get a locally computable PRG that can be used to obtain an MCSP lower bound against $\mathfrak{C}$. Moreover, since the quality of such as a lower bound will depend on the local complexity of the PRG (hence the usefulness of the hard function), we obtain a way of turning average-case hardness against some circuit class $\mathfrak{C}$ into a lower bound for MCSP against the same class, where such a lower bound is quantitatively linked to the average-case hardness.

Combining the above idea with the recent (improved) hardness results for constant-depth circuits augmented with few symmetric gates, due to Servedio and Tan [100], we get MCSP lower bounds against these circuits that match those hardness results (Theorem 1.5).

In Section 6.1 we discuss the Nisan-Wigderson generator and its locality. In Section 6.2, we discuss the framework of proving MCSP lower bounds from average-case hardness and prove Theorem 1.5.

6.1 The Nisan-Wigderson Generator and Its Locality

The idea of using Nisan-Wigderson PRGs to study MCSP and related problems has been explored before (e.g. [6, 87, 47]). However, the previous works were content with the fact that the output of a PRG has circuit complexity at most polynomial in the seed length. Here, we provide a more fine-grained analysis of the local complexity of the Nisan-Wigderson PRG, which depends on the parameters that we choose for the design, and in turn will depend on the "usefulness" of the average-case hard function. This allows us to turn average-case hardness against some circuit class $\mathfrak{C}$ into a lower bound for MCSP against the same class, where such a lower bound is more quantitatively linked to the average-case hardness.

We first review the Nisan-Wigderson framework.

Definition 6.1 (Designs [83])**.** Let N, r, ℓ, α be positive integers. A sequence of sets $S_1, \ldots, S_N$ is a (N, r, ℓ, α)-*design* if

- $\forall j \in [N] : S_j \subseteq [r]$,
- $\forall j \in [N] : |S_j| = \ell$, and
- $\forall j, k \in [s]$, such that $j \neq k$, it is the case that $|S_j \cap S_k| \leq \alpha$.

Lemma 6.2 (Local designs)**.** *For any positive integers N and α, there exists a (N, r, ℓ, α)-design such that $r = N^{2/(\alpha+1)}$ and $\ell = N^{1/(\alpha+1)}$. Moreover, given any $z \in \{0,1\}^r$, and any $j \in \{0,1\}^{\log N}$, $z|_{S_j} \in \{0,1\}^\ell$ can be computed by a circuit of size*

$$O\left(N^{2/(\alpha+1)}\right) + N^{1/(\alpha+1)} \cdot \widetilde{O}(\log N).$$

Proof. Consider the field $\mathbb{F}_\ell$ with ℓ elements. We identify the universe $[r]$ with $\mathbb{F}_\ell \times \mathbb{F}_\ell$ of size ℓ^2. Let $\{e_1, \ldots, e_\ell\}$ be the ℓ elements of the field (in lexicographic order). For each $j \in \{0,1\}^{\log N}$, we view j as an element in $[\ell]^{\alpha+1}$ and identify it with a degree-α polynomial $p_j \in \mathbb{F}_\ell[x]$. Let

$$S_j = \{(e_1, p_j(e_1)), \ldots, (e_\ell, p_j(e_\ell))\}.$$

Note that, for all j, the set S_j is a subset of $\mathbb{F}_\ell \times \mathbb{F}_\ell$, the set S_j has size ℓ, and for two different sets S_j and S_k we have that $|S_j \cap S_k| \leq \alpha$, as the difference $p_j - p_k$ is a polynomial of degree at most α, and thus has at most α roots.

Note that we can hard-wire $\left(e_k, e_k^2, \ldots, e_k^\alpha\right)$ into some circuit, for all $k \in [\ell]$, by using size $\ell \cdot \alpha \cdot \log \ell = \widetilde{O}(\ell)$. Then computing $p_j(e_k)$, for any k, can be done with a circuit of size $\alpha \cdot \widetilde{O}(\log \ell)$ (using Fact 2.3). As a result, S_j can be computed in size

$$\ell \cdot \alpha \cdot \widetilde{O}(\log \ell) = N^{1/(\alpha+1)} \cdot \widetilde{O}(\log N).$$

Once we have the set S_j, we can divide the input z into ℓ equal-size blocks. For each element (a, b) in S_j, we output the b-th bit of the a-th block, using Lemma 4.5, in $O(\ell)$ size. Then, computing $z|_{S_j}$ takes size $\ell \cdot O(\ell) = O\big(N^{2/(\alpha+1)}\big)$. □

Definition 6.3 (Average-case hardness)**.** Let $\mathfrak{C}$ be a class of circuits on N variables. We say that a function f is (s, ε)-*hard against* $\mathfrak{C}$ if, for every $C \in \mathfrak{C}$ of size s, it is the case that

$$\mathop{\mathbf{Pr}}_{x \sim \{0,1\}^N}[f(x) = C(x)] \leq \frac{1}{2} + \varepsilon.$$

Let DNF_α denote the class of DNF circuits on α variables. Note that every α-variate Boolean function can be computed by a DNF of size at most 2^α.

Theorem 6.4 (Nisan-Wigderson generator [83])**.** *Let $\mathfrak{C}$ be a class of circuits on N variables of size s. Let $S_1, \ldots, S_N$ be a (N, r, ℓ, α)-design, and let $f : \{0,1\}^\ell \to \{0,1\}$ be a function that is $(s + N \cdot 2^\alpha, \varepsilon/N)$-hard against $\mathfrak{C} \circ \mathsf{DNF}_\alpha$. Then, the Nisan-Wigderson generator $\mathrm{NW}^f : \{0,1\}^r \to \{0,1\}^N$, defined as*

$$\mathrm{NW}^f(z) = \Big(f\big(z|_{S_1}\big), \ldots, f\Big(z|_{S_N}\Big)\Big),$$

is a PRG that ε-fools $\mathfrak{C}$.

Combining Theorem 6.4 with the design construction in Lemma 6.2, we immediately get the following.

Theorem 6.5 (Local Nisan-Wigderson generator)**.** *Let $s : \mathbb{N} \to \mathbb{N}$ be a function and $\mathfrak{C}$ be a class of circuits on N variables that have size $s = s(N)$. For any $\alpha = \alpha(N)$, if there exists a function $f : \{0,1\}^\ell \to \{0,1\}$, where $\ell = N^{1/(\alpha+1)}$, that is $(s + N \cdot 2^\alpha, 1/(3N))$-hard against $\mathfrak{C} \circ \mathsf{DNF}_\alpha$, then there exists a $\lambda(N)$-local PRG of seed length $N^{2/(\alpha+1)}$ that $(1/3)$-fools $\mathfrak{C}$ with*

$$\lambda(N) = O\Big(N^{2/(\alpha+1)}\Big) + N^{1/(\alpha+1)} \cdot \widetilde{O}(\log N) + \mathrm{CC}(f)\,.$$

We remark that the above local Nisan-Wigderson generator has local complexity that is comparable to its seed length (for this particular local design and modulo the circuit complexity of the hard function).

6.2 MCSP Lower Bounds From Average-case Hard Functions

Next, we demonstrate the use of such local PRGs in obtaining lower bounds for MCSP from average-case hardness results.

One of the restricted circuit classes that have been well studied in circuit complexity is the class of constant-depth circuits augmented with few SYM (symmetric) or THR (linear threshold) gates (see, e.g., [76, 112, 75, 100]). A SYM gate computes a symmetric function, which is a Boolean function whose output depends only on the sum of its input variables. A THR gate computes a linear threshold function, which is a Boolean function defined as the sign of some linear form, over Boolean variables, with real coefficients.

We will combine the above local Nisan-Wigderson framework with the following average-case lower bounds against the class of constant-depth circuits augmented with a few (sublinearly many) symmetric and linear threshold gates.

Theorem 6.6 ([100, Theorem 4])**.** *There exists a constant $\tau > 0$ such that the following holds. For any ℓ, there exists a function $f : \{0,1\}^{\ell} \to \{0,1\}$ that is $\left(\ell^{\tau \log \ell}, \exp\left(-\Omega(\ell^{0.499})\right)\right)$-hard against AC^0 circuits of size $\ell^{\tau \log \ell}$ with at most $\ell^{0.249}$ SYM or THR gates. Moreover, f can be computed by a circuit of size $O(\ell)$.*

As a result, we get a local PRG that fools such circuits.

Corollary 6.7. *There exists some constant $\tau > 0$ such that, for any $s : \mathbb{N} \to \mathbb{N}$ that satisfies $s(N) \geq N$ for all $N \in \mathbb{N}$, there exists a $\lambda(N)$-local PRG that $(1/3)$-fools AC^0 circuits of*

size $s(N) \leq \ell^{\tau \log \ell}$*, for some* $\ell = \ell(N) > 0$*, with at most* $\ell^{0.249}$ SYM *or* THR *gates and* $\lambda(N) = 2^{O(\sqrt{\log s(N)})}$.

Proof. Let $\mathfrak{C}$ be the class of AC^0 circuits of size $\ell^{\tau \log \ell}$, for some constant $\tau > 0$, with at most $\ell^{0.249}$ SYM or THR gates. Let $N \in \mathbb{N}$ be sufficiently large and $s := s(N)$. Choose

$$\alpha = \tau' \cdot \frac{\log N}{\sqrt{\log s}},$$

where $\tau' > 0$ is some sufficiently small constant. Then, for $\ell = N^{1/(\alpha+1)}$, if we can show the existence of some efficiently computable function $f : \{0,1\}^{\ell} \to \{0,1\}$ that is $(s + N \cdot 2^{\alpha}, 1/(3N))$-hard against $\mathfrak{C} \circ \mathsf{DNF}_{\alpha}$, then the result follows from Theorem 6.5. The existence of such a function is given by Theorem 6.6, by noting that for our choice of α we have

$$\ell^{\tau \log \ell} \geq s + N \cdot 2^{\alpha},$$

and

$$\exp\left(-\Omega(\ell^{0.249})\right) \leq 1/(3N). \qquad \square$$

We remark that the above example does not take advantage of the fact that the local complexity of the Nisan-Wigderson PRG is almost the same as its seed length. This is because, in this case, the seed length has some arbitrary constant in the exponent.

Combining Corollary 6.7 with Theorem 3.4, we get the following.

Theorem 6.8 (Theorem 1.5, restated)**.** *There exists a constant* $\gamma > 0$ *such that the following holds. Let* $\mathfrak{C}$ *be the class of constant-depth* N*-input* AC^0 *circuits augmented with at most* $2^{\gamma\sqrt{\log N}}$ SYM *or* THR *gates. Then, any circuit in* $\mathfrak{C}$ *computing* MCSP *on truth tables of length* N *must have size* $N^{\Omega(\log N)}$.

Remark 6.9 (Can we improve the lower bound of Theorem 6.8?). The bottleneck in improving Theorem 6.8 lies at the existing average-case lower bounds against AC^0 circuits augmented

with SYM or THR gates. This is so, because the proof of Theorem 6.8 makes use of the locality of the Nisan-Wigderson PRG, when instantiated with the hard function f of Servedio and Tan [100] (see Theorem 6.6), and the locality of the Nisan-Wigderson PRG depends on the average-case hardness of f. The latter becomes evident when ones inspects Theorem 6.5: If the function f used in instantiating Nisan-Wigderson is hard (in the average-case sense), then the parameter α in Theorem 6.5 will be large, by Definition 6.3, and so the locality λ from Theorem 6.5 will become small (because it is proportional to $1/\alpha$). By the discussion of Chapter 3 ("MCSP lower bounds from local PRGs"), this would result in a strong lower bound against AC^0 circuits augmented with SYM or THR gates. (For a concrete example on how the locality of a PRG directly affects the resulting MCSP lower bound, please see the proof of Theorem 4.6.)

Notes

The results of this chapter are products of joint work with Mahdi Cheraghchi, Valentine Kabanets, and Zhenjian Lu [29].

Chapter 7

MCSP Lower Bounds Against De Morgan Formulas of Parities

Although there are well-known PRG constructions in the literature that fool subcubic-size De Morgan formulas [57], such a construction was missing for subcubic-size De Morgan formulas of parities (i.e., the class $\mathsf{FORMULA} \circ \mathsf{XOR}$). In this chapter, we present such a construction. In particular, we show that there is a PRG of seed length $O(\sqrt{s} \cdot \log s \cdot \log(1/\varepsilon) + \log n)$ that ε-fools $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ (Theorem 1.7). As a result, by the discussion of Section 3.2, MCSP is not in $\mathsf{FORMULA}[n^{1.99}] \circ \mathsf{XOR}$ (Theorem 1.8).

In order to explain our techniques, we focus on the design of PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$. The MCSP lower bounds will then follow by the discussion of Section 3.2.

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

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

The only known proof of this result [95] relies on a sequence of works [16, 66, 51, 35, 94, 14, 96] on quantum query complexity, generalizing Grover's search algorithm for the OR predicate [42] to arbitrary formulas. A consequence of **(A)** which is implicit in the work of Tal [106] is the following.

> **(B)** Let $\mathcal{D}$ be a distribution over $\{0,1\}^m$, and $F \in \mathsf{FORMULA}[s] \circ \mathcal{G}$. Then, for every function f,
>
> $$\text{if } \Pr_{x\sim\mathcal{D}}[F(x) = f(x)] \geq 1/2 + \varepsilon, \text{ then } \Pr_{x\sim\mathcal{D}}[h(x) = f(x)] \geq 1/2 + \exp(-t),$$
>
> for some function h which is the XOR of at most t functions in $\mathcal{G}$, where $t = \widetilde{\Theta}(\sqrt{s} \cdot \log(1/\varepsilon))$.

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

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

In what follows, Section 7.1 contains some background material. In Section 7.2, we describe a general approach of constructing PRGs for $\mathsf{FORMULA} \circ \mathcal{G}$. Finally, in Section 7.3, we prove Theorem 1.7 and Theorem 1.8.

7.1 Preliminaries

7.1.1 Notation

In this chapter, we will use $\{-1, 1\}$ as the Boolean basis, where we identify -1 with TRUE and 1 with FALSE.

7.1.2 Augmented De Morgan Formulas

In this chapter, we denote by $\mathsf{FORMULA}[s]$ the class of Boolean functions computable by size-s De Morgan formulas. Let $\mathcal{G}$ denote some class of Boolean functions. Then, we denote by $\mathsf{FORMULA}[s] \circ \mathcal{G}$ the class of languages computable by some size-s De Morgan formula where its leaves are labelled by functions in $\mathcal{G}$. For example, $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ denotes size-s De Morgan formulas of parities.

7.1.3 Approximating Polynomials

The following will come in handy in the proof of Theorem 1.7.

Definition 7.1 (Point-wise approximation)**.** For a Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$, we say that the function $\widetilde{f} : \{-1, 1\}^n \to \mathbb{R}$ *ε-approximates* f if for every $z \in \{-1, 1\}^n$,

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

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

Theorem 7.2 ([95], see also [22])**.** *Let $s > 0$ be an integer and $0 < \varepsilon < 1$. Any De Morgan formula $F : \{-1, 1\}^n \to \{-1, 1\}$ of size s has a ε-approximating polynomial of degree $d =$*

$O(\sqrt{s} \cdot \log(1/\varepsilon))$. *That is, there exists a degree-d polynomial $p : \{-1,1\}^n \to \mathbb{R}$ over the reals such that for every $z \in \{-1,1\}^n$,*

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

7.1.4 Fourier basis of Boolean functions

For any Boolean function $f : \{-1,1\}^n \to \{-1,1\}$ and any $S \subseteq [n]$ there exists a number $\widehat{f}(S) \in \mathbb{R}$ such that

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \prod_{i \in S} x_i$$

for any $x = (x_1, \ldots, x_n) \in \{-1,1\}^n$.

7.2 Pseudorandom Generators: Our Framework

Our PRG is obtained from a general framework that allows us to reduce the task of fooling $\mathsf{FORMULA} \circ \mathcal{G}$ to the task of fooling the class of functions that are parities of a few functions from $\mathcal{G}$.

Theorem 7.3 (PRG for $\mathsf{FORMULA} \circ \mathcal{G}$ from PRG for $\mathsf{XOR} \circ \mathcal{G}$)**.** *There exists a constant $c > 0$ such that the following holds. Let $\mathcal{G}$ be a class of gates on n bits. For any integer $s > 0$ and any $0 < \varepsilon < 1$, if a distribution $\mathcal{D}$ over $\{-1,1\}^n$ $\left(2^{-c \cdot \sqrt{s} \cdot \log(s) \cdot \log(1/\varepsilon)}\right)$-fools the XOR (parity) of $c \cdot \sqrt{s} \cdot \log(1/\varepsilon)$ arbitrary functions from $\mathcal{G}$, then $\mathcal{D}$ also ε-fools $\mathsf{FORMULA}[s] \circ \mathcal{G}$.*

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

$$\mathbf{E}[C(\mathcal{D})] \overset{\varepsilon}{\approx} \mathbf{E}[C(U)]. \tag{7.1}$$

Let p be a $(\varepsilon/3)$-approximating polynomial for F given by Theorem 7.2. Note that the degree of p is

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

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

$$\widetilde{C} := p(g_1, g_2 \ldots, g_s).$$

Since $\widetilde{C}$ approximates C in a point-wise fashion, we have

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

and

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

Then to show Equation (7.1), it suffices to show

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

We have

$$\begin{aligned}\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[\widetilde{C}(x)] &= \mathop{\mathbf{E}}_{x\sim D}\left[\sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot\prod_{i\in S} g_i(x)\right] \\ &= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S)\cdot \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\prod_{i\in S} g_i(x)\right]. \end{aligned} \tag{7.2}$$

Now note that for each $S \subseteq [s]$, $\prod_{i\in S} g_i(x)$ computes the XOR of at most d functions from $\mathcal{G}$.

Using the fact the distribution $\mathcal{D}$ $\left(\delta = 1/2^{c\cdot\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon)}\right)$-fools the XOR of any d functions from $\mathcal{G}$, we get

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

$$= \sum_{\substack{S\subseteq[s]:\\ |S|\leq d}} \hat{p}(S) \cdot \left(\mathop{\mathbf{E}}_{x\sim U} \left[\prod_{i\in S} g_i(x) \right] + \delta_S \right),$$

where $|\delta_S| \leq \delta$, or

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

It remains to show

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

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

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

Then,

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

where the last inequality holds for some sufficiently large constant c. This concludes the proof. □

7.3 Applications and MCSP Lower Bounds: Fooling De Morgan Formulas of Parities

Theorem 7.4 (Theorem 1.7, restated)**.** *For any $s > 0$ and $0 < \varepsilon < 1$, there exists a PRG that ε-fools n-input formulas of parities from* $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$ *with seed length*

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

Proof. By Theorem 7.3, to fool $\mathsf{FORMULA}[s] \circ \mathcal{G}$ it suffices to $\left(\delta = 1/2^{O\left(\sqrt{s}\cdot\log(s)\cdot\log(1/\varepsilon)\right)}\right)$-fool the XOR of a few functions from $\mathcal{G}$, where $\mathcal{G}$ is the class of XOR functions. Note that the XOR of any set of XOR functions simply computes some XOR function. Therefore, we can use a small-biased distribution (which by definition fools every XOR function) to fool $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$.

Finally, note that there are known constructions of δ-biased distributions that use $O(\log(n/\delta))$ random bits (see Theorem 2.10). □

Using the locality of this PRG for $\mathsf{FORMULA} \circ \mathsf{XOR}$ (see Theorem 2.10) we get a lower bound for MCSP against subquadratic-size formulas of XORs, by Theorem 3.4.

Theorem 7.5 (Theorem 1.8, restated)**.** *For every $s > 0$, if $\mathrm{MCSP}\left[2^{\Omega(n)}\right]$ on N-bit truth tables can be computed by a formula of parities in* $\mathsf{FORMULA}[s] \circ \mathsf{XOR}$*, then $s = \widetilde{\Omega}(N^2)$.*

Notes

The results of this chapter are products of joint work with Valentine Kabanets, Sajin Koroth, Zhenjian Lu, and Igor C. Oliveira [63].

Chapter 8

MCSP Lower Bounds Against One-tape Turing Machines

As mentioned in Chapter 1, a recent line of work has exhibited hardness magnification phenomena for MCSP in the sense that a very weak lower bound for MCSP implies a breakthrough result in complexity theory. For example, McKay, Murray, and Williams (STOC 2019) implicitly showed that, for some constant $\mu_1 > 0$, if MCSP$[2^{\mu_1 \cdot n}]$ cannot be computed by a one-tape Turing machine (with an additional one-way read-only input tape) running in time $N^{1.01}$, then $\mathsf{P} \neq \mathsf{NP}$ (Theorem 1.9). A natural question arises:

> How easy is it to prove that MCSP$[2^{\mu_1 \cdot n}]$ cannot be computed by a one-tape Turing machine (with an additional one-way read-only input tape) running in time $N^{1.01}$?

In this chapter, we present the following new lower bound against one-tape Turing machines (see Theorem 1.10 and Theorem 1.11), that is related to the question above (and deceivingly implies that the question posed above can be answered in the affirmative):

> A randomized two-sided error one-tape Turing machine (with an additional one-way read-only input tape) cannot compute MCSP$[2^{\mu_2 \cdot n}]$ in time $N^{1.99}$, for some constant $\mu_2 > \mu_1$.

As one may observe, our lower bound is much stronger than what would suffice to separate P from NP, but refers to a parameterized version of MCSP that has a *larger* size parameter. This does not allow us to conclude that $\mathsf{P} \neq \mathsf{NP}$ by simply invoking the result by McKay, Murray, and Williams stated above.

However, our result is the first non-trivial lower bound for MCSP against one-tape Turing machines, and essentially matches the best-known lower bounds for any explicit function against this computational model. It is based on recent constructions of pseudorandom generators for read-once oblivious branching programs (ROBPs) and combinatorial rectangles [37, 113]. Details follow.

Let $\mathfrak{C}$ be a circuit class. As we saw in Chapter 3, a general approach for obtaining a $\mathfrak{C}$-lower bound for MCSP is by constructing a local pseudorandom generator (PRG) that fools $\mathfrak{C}$. We invoke this framework, as outlined below.

At the core of our argument is the recent breakthrough result of Forbes and Kelley [37], who constructed the first pseudorandom generator with $\mathsf{polylog}(n)$ seed length that fools any-order read-once oblivious branching programs. Viola [113] used their construction to obtain a pseudorandom generator that fools *deterministic* Turing machines (DTMs). Herein, we generalize his result to the case of *randomized* Turing machines (RTMs).

At a high level, our crucial idea is that Viola's proof does not exploit the uniformity of Turing machines, and hence a good coin flip sequence of a randomized oracle algorithm and all of its oracle queries and corresponding answers can be fixed as non-uniformity (Lemma 8.11). In addition, by a careful examination of the Forbes-Kelley PRG, we show that their PRG is

local. This observation gives rise to a local PRG that fools $\mathsf{BPTIME}_1[t(N)]$, which yields a proof of Theorem 1.11.

In Section 8.1, we give the necessary background. We prove Theorem 1.9 in Section 8.2. The main ideas of Theorem 1.11 are described in Section 8.3. In Section 8.4 we show that the pseudorandom generators that we use are local, which will complete the proof of Theorem 1.11.

8.1 Preliminaries

8.1.1 One-tape Turing Machines

Throughout this chapter, we consider a Turing machine that has one work tape and a one-way input tape. In this context, "one-way" means that the tape-head may move only from left to right.

A *deterministic Turing machine (DTM)* is a Turing machine with two tapes: A two-way read/write work tape and a one-way read-only input tape. Let $x \in \{0,1\}^*$ and M be a DTM; we write $M(x)$ to denote the output of M when its input tape is initialized with x and its work tape is empty. Let $t : \mathbb{N} \to \mathbb{N}$ be time-constructible. The class of languages $L \subseteq \{0,1\}^*$ decided by some $O(1)$-state time-t DTM is denoted by $\mathsf{DTIME}_1[t]$.

We also consider a randomized variant of DTMs. A *randomized Turing machine (RTM)* is a Turing machine with three tapes: A two-way read/write work tape, a one-way read-only input tape, and a one-way read-only random tape. Let $x, r \in \{0,1\}^*$ and M be a RTM; we write $M(x,r)$ to denote the output of M when its input tape contains x, its work tape is empty, and its random tape contains r. Let $t : \mathbb{N} \to \mathbb{N}$ be time-constructible. For a language $L \subseteq \{0,1\}^*$ and an RTM M, we say that M *decides* L *with two-sided error* if $\Pr_r[M(x,r) = 1] \geq \frac{2}{3}$ for every input $x \in L$ and $\Pr_r[M(x,r) = 0] \geq \frac{2}{3}$ for every input $x \notin L$. The class of languages $L \subseteq \{0,1\}^*$

decided by some $O(1)$-state time-t RTM with two-sided error is denoted by $\mathsf{BPTIME}_1[t]$.

A *randomized oracle Turing machine (oracle RTM)* is a Turing machine with four tapes: A two-way read/write work tape, a one-way read-only input tape, a one-way read-only random tape, and an oracle tape. This model is identical to the randomized Turing machine model apart from the oracle tape, which is a standard oracle tape. The class of languages $L \subseteq \{0,1\}^*$ decided by some $O(1)$-state time-t oracle RTM, with oracle access to some $O \subseteq \{0,1\}^*$ and two-sided error, is denoted by $\mathsf{BPTIME}_1^O[t]$.

8.1.2 Streaming Algorithms

A *space-$s(n)$ streaming algorithm* with update time $u(n)$ on an input $x \in \{0,1\}^n$ has a working storage of $s(n)$ bits. At any point, the algorithm can either choose to perform one operation on $O(1)$ bits in storage or it can choose to read the next bit from the input. The total time between two next-bit reads is at most $u(n)$ and the final outcome is reported in $O(u(n))$ time.

Lemma 8.1. *Any one-pass streaming algorithm with $t(N)$ update time, on inputs of length N, can be simulated by a one-tape Turing machine with a one-way read-only input tape running in time $O(N \cdot \mathrm{poly}(t(N)))$.*

Proof. Recall that a streaming algorithm reads one bit of its input from left to right, and each consecutive read operation occurs within $t(N)$ steps. By Cook and Reckhow [30], $\mathrm{poly}(t(N))$ steps suffice for some multi-tape Turing machine to perform an update. Since for any time constructible function $T : \mathbb{N} \to \mathbb{N}$ a one-tape Turing machine can simulate a $T(n)$-time multi-tape Turing machine within $O(T(n)^2)$ steps, $\mathrm{poly}(t(N))^2 = \mathrm{poly}(t(N))$ steps suffice for a one-tape Turing machine to simulate an update. Thus, it $N \cdot \mathrm{poly}(t(N))$ steps in total suffice to finish the computation on inputs of length N in the one-tape Turing machine model, that is, a streaming algorithm can be simulated in time $N \cdot \mathrm{poly}(t(N))$ by a one-tape Turing machine. □

8.1.3 Pseudorandom Generators That Fool Oracle RTMs

For our purpose, it is useful to extend the notion of PRG to fools randomized oracle (one-tape) Turing machines.

Definition 8.2. For functions $s, t : \mathbb{N} \to \mathbb{N}$ such that $s(n) < n$ for all $n \in \mathbb{N}$, $q > 0$, a language $O \subseteq \{0,1\}^*$, and a function $\varepsilon : \mathbb{N} \to (0,1)$, a function $\left\{G_n : \{0,1\}^{s(n)} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$ is a *pseudorandom generator that ε-fools q-state time-t RTMs with oracle O* if

$$\left| \Pr_{\substack{x\sim\{0,1\}^n,\\ r\sim\{0,1\}^{t(n)}}} \left[M^O(x,r)=1\right] - \Pr_{\substack{y\sim\{0,1\}^{s(n)},\\ r\sim\{0,1\}^{t(n)}}} \left[M^O(G_n(y),r)=1\right] \right| \leq \varepsilon(n)$$

for any q-state time-t RTM M with oracle O.

8.1.4 MCSP Lower Bounds Against Oracle RTMs, From Local PRGs

We observe that a local pseudorandom generator for oracle RTMs yields MCSP lower bounds (similarly to what we saw in Section 3.2 and Theorem 3.4).

Theorem 8.3 (A randomized analogue of Theorem 3.4)**.** *Let $t : \mathbb{N} \to \mathbb{N}$ and $0 < \mu < 1$ be such that* $\mathrm{MCSP}[2^{\mu\cdot n}]$ *on truth tables of length $N := 2^n$ can be computed by a time-$t(N)$ oracle RTM. Then, if for a function $\lambda : \mathbb{N} \to \mathbb{N}$ there exists a $\lambda(N)$-local PRG G that $(1/6)$-fools time-$t(N)$ oracle RTMs, then $\lambda(N) \geq 2^{\mu\cdot n}$.*

Proof. Let M be a time-$t(N)$ oracle RTM that computes $\mathrm{MCSP}[2^{\mu\cdot n}]$ on truth tables of length N. Let G be a $\lambda(N)$-local PRG that $(1/6)$-fools M.

For the sake of contradiction, suppose that $\lambda(N) < 2^{\mu\cdot n}$. On the one hand, since most functions require circuits of size greater than $N/(3\log N) = 2^n/(3n) > 2^{\mu\cdot n}$ (Theorem 2.2)

and M computes MCSP, we have

$$\mathop{\mathbf{Pr}}_{w\sim\{0,1\}^N}[w \notin \mathrm{MCSP}[2^{\mu\cdot n}]] \geq 1 - o(1)$$

or

$$\mu := \mathop{\mathbf{Pr}}_{w,r}[M(w,r) = 0] \geq 1 - o(1) - 1/3 > 1/2.$$

Also, since G $(1/6)$-fools M, we have

$$\mathop{\mathbf{Pr}}_{z,r}[M(G(z),r) = 0] \geq \mu - 1/6 > 1/3.$$

On the other hand, since G is $\lambda(N)$-local, we must have

$$\mathop{\mathbf{Pr}}_{z,r}[M(G(z),r) = 1] \geq 2/3$$

for every z. This gives a contradiction. □

8.2 Hardness Magnification for One-tape Turing Machines

In this section, we obtain the following hardness magnification result for one-tape Turing machines.

Theorem 8.4 (A corollary of McKay, Murray, and Williams [77]; Theorem 1.9, restated)**.** *There exists a constant $\mu > 0$ such that, if $\mathrm{MCSP}[2^{\mu n}]$ is not in $\mathsf{DTIME}_1\left[N^{1.01}\right]$, then $\mathsf{P} \neq \mathsf{NP}$.*

Proof. McKay, Murray, and Williams [77, Theorem 1.3] showed that if $\mathsf{P} = \mathsf{NP}$, then there exists a polynomial p such that, for any time-constructible function $s(n)$, there exists a one-pass streaming algorithm with update time $p(s(n))$ that computes $\mathrm{MCSP}[s(n)]$. By Lemma 8.1, we obtain $\mathrm{MCSP}[s(n)] \in \mathsf{DTIME}_1[N \cdot p(s(n))]$, where $N = 2^n$. Depending on p, we choose a small constant $\mu > 0$ and set $s(n) := 2^{\mu n}$ so that $N \cdot p(s(n)) = N^{1+O(\mu)} \leq N^{1.01}$.

To summarize, we have proved that if $\mathsf{P} = \mathsf{NP}$, then for some constant $\mu > 0$, $\mathrm{MCSP}[2^{\mu n}] \in \mathsf{DTIME}_1[N^{1.01}]$. This statement is logically equivalent to the following: There exists a constant $\mu > 0$ such that $\mathsf{P} = \mathsf{NP}$ implies that $\mathrm{MCSP}[2^{\mu n}] \notin \mathsf{DTIME}_1[N^{1.01}]$ (because the statement that $\mathsf{P} = \mathsf{NP}$ is independent of μ). Taking its contrapositive, we obtain the desired result. □

8.3 MCSP Lower Bounds Against One-tape Oracle RTMs

In this section, we present a proof of our main result.

Theorem 8.5 (Theorem 1.11, restated)**.** *Let $O \subseteq \{0,1\}^*$ be any language. Then, for every constant $1/2 < \mu < 1$, $\mathrm{MCSP}[2^{\mu \cdot n}]$ on truth tables of size $N := 2^n$ is not in $\mathsf{BPTIME}_1^O\left[N^{2\cdot(\mu' - o(1))}\right]$ for all $1/2 < \mu' < \mu$, where all of the strings queried to O are of length $N^{o(1)}$.*

8.3.1 Connections to Hardness Magnification

Theorem 8.5 implies that establishing hardness magnification phenomena for MCSP, when the circuit size threshold parameter is $2^{0.51n}$, would require the development of new techniques; see Remark 8.8. To explain why this is true, we shall first require the following result by McKay, Murray, and Williams [77] that gives an oracle streaming algorithm for MCSP.

Lemma 8.6 ([77, Theorem 1.2])**.** *Let $s : \mathbb{N} \to \mathbb{N}$ be a size function, with $s(n) \geq n$ for all n, and $N := 2^n$. Then, there is a one-pass streaming algorithm for $\mathrm{MCSP}[s(n)]$ on N-bit inputs running in $N \cdot \widetilde{O}(s(n))$ time with $\widetilde{O}\left(s(n)^2\right)$ update time and $\widetilde{O}(s(n))$ space, using an oracle for $\Sigma_3\mathrm{SAT}$ with queries of length $\widetilde{O}(s(n))$.*

A corollary of Lemma 8.6 and Lemma 8.1 is the following.

Corollary 8.7 (Consequences of hardness magnification from currently known techniques)**.** *Let $s : \mathbb{N} \to \mathbb{N}$ be a size function. Then, $\mathrm{MCSP}[s(n)]$ on truth tables of length $N := 2^n$ is in*

$\mathsf{DTIME}_1^O[N \cdot \mathrm{poly}(s(n))]$, *for some* $O \in \Sigma_3^{\mathsf{P}}$, *where all of the strings queried to* O *are of length at most* $\mathrm{poly}(s(n))$.

The following remark summarizes the main idea of this subsection.

Remark 8.8. By Corollary 8.7, we see that if $s(n) = 2^{\mu \cdot n}$, for $\mu = o(1)$, then MCSP$[s(n)]$ is in $\mathsf{DTIME}_1^O\left[N^{1+o(1)}\right]$, where all of the strings queried to O are of length $N^{o(1)}$. In light of this observation, Theorem 8.5 is important for the following reason. As $\mathsf{DTIME}_1^O\left[N^{1+o(1)}\right]$ is a subset of $\mathsf{BPTIME}_1^O\left[N^{2\cdot(\mu'-o(1))}\right]$ for all $1/2 < \mu' < 1$ and all languages $O \subseteq \{0,1\}^*$, Theorem 8.5 shows that establishing hardness magnification phenomena for MCSP$[s(n)]$ like that of Theorem 1.9, when $s(n) = 2^{\mu \cdot n}$ for any constant $1/2 < \mu < 1$, would require the development of techniques that do not rely on designing oracle algorithms that make short oracle queries.

Comparison With the Locality Barrier

Chen, Hirahara, Oliveira, Pich, Rajgopal, and Santhanam [25] introduced the "locality barrier" to explain why it will be difficult to acquire a major complexity breakthrough through the lens of hardness magnification. Their reasoning goes as follows:

> Existing magnification theorems unconditionally show that problems, against which some circuit lower bound implies a complexity-theoretic breakthrough, admit highly efficient small fan-in oracle circuits, while lower bound techniques against weak circuit models quite often easily extend to circuits containing such oracles.

Our Remark 8.8, therefore, is close in spirit to the results of Chen, Hirahara, Oliveira, Pich, Rajgopal and Santhanam [25]: We make use of a lower bound (Theorem 8.5) to motivate the development of new techniques for proving hardness magnification phenomena while

Chen, Hirahara, Oliveira, Pich, Rajgopal and Santhanam make use of hardness magnification phenomena to motivate the development of new techniques for acquiring lower bounds; a notable difference is that we consider one-tape Turing machines while they consider Boolean circuits.

8.3.2 Proof of Theorem 8.5

In order to prove Theorem 8.5, our goal is to construct a local pseudorandom generator that fools oracle RTMs and then apply Theorem 8.3. Viola [113] constructed a pseudorandom generator that fools the one-tape Turing machine model (DTM).[1] We will show that, in fact, the same construction fools oracle RTMs as well. In order to do so, we recall the idea of Viola [113]. The idea is that, in order to fool DTMs, it is sufficient to use a PRG that ε-fools ROBPs for an exponentially small ε. This is because time-t DTMs can be written as the sum of an exponential number of ROBPs.

Lemma 8.9 (Viola [113])**.** *Let $n \in \mathbb{N}$ and M be a q-state time-t DTM. Then, there is a family $\{P_\alpha\}_{\alpha \in A}$ of n-input ROBPs of width $\exp\big(O\big(\sqrt{t} \cdot \log(tq)\big)\big)$ such that, for any $x \in \{0,1\}^n$,*

$$M(x) = \sum_{\alpha \in A} P_\alpha(x),$$

where $|A| \leq (tq)^{O(\sqrt{t})}$.

By a simple calculation (cf. Section 8.4), any pseudorandom generator that $\varepsilon/|A|$-fools ROBPs also ε-fools DTMs. Viola [113] then used the pseudorandom generator of Forbes and Kelley [37] that fools ROBPs. By a careful examination, we will show that the Forbes-Kelley pseudorandom generator is local; see Corollary 8.19 in Section 8.4.

[1] We note that our definition of PRG is different from that of [113] in that a random tape is not regarded as an input tape.

Theorem 8.10 (Forbes-Kelley PRG is local)**.** *There exists a* $\widetilde{O}\big((\sqrt{t}+\log(1/\varepsilon))\cdot\log q\big)\cdot$ polylog(n)*-local pseudorandom generator with seed length* $\widetilde{O}\big((\sqrt{t}+\log(1/\varepsilon))\cdot\log q\big)$ *that* ε*-fools* q*-state time-*t n*-input DTMs for any* $t\geq n$*.*

Our main idea for obtaining an oracle randomized Turing machine lower bound is that Viola's reduction can be applied to *non-uniform computational models*, i.e., q-state Turing machines where q can become large as the input length becomes large. More specifically, it is possible to incorporate all possible oracle queries (along with their answers) and any good coin flip sequence r into the internal states of DTMs.

Lemma 8.11. *For an input length* $n\in\mathbb{N}$*, for any* q*-state time-*t *oracle RTM* M*, that only queries strings of length at most* ℓ *to its oracle* O*, and a coin flip sequence* $r\in\{0,1\}^t$*, there exists some* $\big(q\cdot 2^\ell\cdot t\big)$*-state time-*$t$ *DTM* M' *such that* $M'(x)=M^O(x,r)$ *for every input* $x\in\{0,1\}^n$*.*

Proof. Let Q_M denote the set of the states of M. We define the set of the states of M' as

$$Q_{M'}:=\Big\{(q,s,b,i)\in Q_M\times\{0,1\}^\ell\times\{0,1\}\times[t]\mid O(s)=b\Big\}.$$

The transition from the state $(q,s,b,i)\in Q_{M'}$ can be defined in a natural way, by using the i-th bit of r, namely r_i, the state q, and the fact that $O(s)=b$. □

Corollary 8.12. *There exists a* $\widetilde{O}\big((\sqrt{t}+\log(1/\varepsilon))\cdot\log(q\cdot 2^\ell\cdot t)\big)\cdot$polylog$(n)$*-local pseudorandom generator with seed length*

$$\sigma(t,q,\varepsilon)=\widetilde{O}\Big((\sqrt{t}+\log(1/\varepsilon))\cdot\log(q\cdot 2^\ell\cdot t)\Big)$$

that ε*-fools* q*-state time-*t n*-input oracle RTMs that may only query strings of length at most* ℓ *to their oracle, for any* $t\geq n$*.*

Proof. We hard-code the oracle queries and their answers in the internal states and, moreover, we use an averaging argument to fix one good coin flip sequence r. Let M be any q-state time-t oracle RTM that may query to its oracle O strings of length at most ℓ. Let G be a PRG from Theorem 8.10. We have that

$$\begin{aligned}
&\left| \mathop{\mathbf{Pr}}_{\substack{r\sim\{0,1\}^t,\\ x\sim\{0,1\}^n}}\big[M^O(x,r)=1\big] - \mathop{\mathbf{Pr}}_{\substack{r\sim\{0,1\}^t,\\ y\sim\{0,1\}^{\sigma(t,q,\varepsilon)}}}\big[M^O(G(y),r)=1\big] \right| \\
&\quad= \left| \mathop{\mathbf{E}}_{r\sim\{0,1\}^t}\left[\mathop{\mathbf{E}}_{x\sim\{0,1\}^n}\big[M^O(x,r)\big]\right] - \mathop{\mathbf{E}}_{r\sim\{0,1\}^t}\left[\mathop{\mathbf{E}}_{y\sim\{0,1\}^{\sigma(t,q,\varepsilon)}}\big[M^O(G(y),r)\big]\right] \right| \\
&\quad= \left| \mathop{\mathbf{E}}_{r}\left[\mathop{\mathbf{E}}_{x}\big[M^O(x,r)\big] - \mathop{\mathbf{E}}_{y}\big[M^O(G(y),r)\big]\right] \right| \\
&\quad\le \mathop{\mathbf{E}}_{r}\left[\left|\mathop{\mathbf{E}}_{x}\big[M^O(x,r)\big] - \mathop{\mathbf{E}}_{y}\big[M^O(G(y),r)\big]\right|\right] \\
&\quad\le \left|\mathop{\mathbf{E}}_{x}\big[M^O(x,r^*)\big] - \mathop{\mathbf{E}}_{y}\big[M^O(G(y),r^*)\big]\right|,
\end{aligned}$$

for some $r^* \in \{0,1\}^t$, by an averaging argument. By applying Lemma 8.11, for M, O, and r^*, we obtain an equivalent $\big(q\cdot 2^\ell \cdot t\big)$-state time-$t$ DTM M'. The result now follows from Theorem 8.10. Specifically,

$$\begin{aligned}
\left|\mathop{\mathbf{E}}_{x}\big[M^O(x,r^*)\big] - \mathop{\mathbf{E}}_{y}\big[M^O(G(y),r^*)\big]\right| &= \left|\mathop{\mathbf{E}}_{x}\big[M'(x)\big] - \mathop{\mathbf{E}}_{y}\big[M'(G(y))\big]\right| \\
&= \left|\mathop{\mathbf{Pr}}_{x}\big[M'(x)=1\big] - \mathop{\mathbf{Pr}}_{y}\big[M'(G(y))=1\big]\right| \le \varepsilon. \qquad \square
\end{aligned}$$

We now prove Theorem 8.5.

Proof of Theorem 8.5. Take the pseudorandom generator G of Corollary 8.12 with parameter $\varepsilon := 1/6$. Let $1/2 < \mu' < \mu < 1$ be arbitrary constants. Let $t, \ell : \mathbb{N} \to \mathbb{N}$ be functions such that $t(N) = N^{2\cdot(\mu' - o(1))}$ and $\ell(n) = 2^{o(n)}$. Then, the circuit complexity of the Boolean functions encoded by the outputs of G (that is, the locality of G) is at most

$$\widetilde{O}\Big(\sqrt{t(N)}\cdot(\log q + \ell(n))\Big)\cdot \mathrm{polylog}(N) \le \widetilde{O}(N^{\mu'-o(1)+o(1)+o(1)}) < 2^{\mu\cdot n},$$

where $N = 2^n$. By the contrapositive of Theorem 8.3, we obtain that $\mathrm{MCSP}[2^{\mu \cdot n}]$ is not in $\mathsf{BPTIME}_1^O[t(N)]$, where all of the strings queried to O are of length $2^{o(n)} = N^{o(1)}$. □

8.4 The Forbes-Kelley PRG is Local

In this section, we show that the Forbes-Kelley PRG is local. That is, we prove Theorem 8.10.

8.4.1 γ-almost k-wise Independent Distributions

Below, we define γ-almost k-wise independent distributions.

Definition 8.13. Let $n \in \mathbb{N}$, $0 < \gamma < 1$, and $k > 0$. A random variable $X = (x_1, \ldots, x_n)$ over $\{0,1\}^n$ is said to be *γ-almost k-wise independent with respect to the distribution D* if, for any k indices $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, it is the case that

$$\sum_{\alpha \in \{0,1\}^k} \left| \mathop{\mathbf{Pr}}_{X \sim D}[x_{i_1} \cdots x_{i_k} = \alpha] - \mathop{\mathbf{Pr}}_{X \sim \{0,1\}^n}[x_{i_1} \cdots x_{i_k} = \alpha] \right| \leq \gamma.$$

On this occasion, we also say that D is a *γ-almost k-wise independent distribution over* $\{0,1\}^n$.

We present a δ-almost k-wise independent distribution, and an efficient δ-almost k-wise independent generator.

Lemma 8.14 ([81])**.** *Let $0 < \delta < 1$ and D be an δ-biased distribution over n-bit strings. Then, for any $k \in \mathbb{N}$, D is a $\left(2^{k/2} \cdot \delta\right)$-almost k-wise independent distribution over n-bit strings.*

Theorem 8.15. *There exists a $\widetilde{O}(k + \log(n/\delta)) \cdot \mathrm{polylog}(n)$-local δ-almost k-wise independent generator $G : \{0,1\}^s \to \{0,1\}^n$ with seed length $s = O(k + \log(n/\delta))$.*

Proof. By setting $\varepsilon := 2^{-k/2} \cdot \delta$, the ε-biased pseudorandom generator of Theorem 2.10 is a $\widetilde{O}(k + \log(n/\delta)) \cdot \mathrm{polylog}(n)$-local δ-almost k-wise independent generator (by Lemma 8.14). □

8.4.2 The Forbes-Kelley PRG

Definition 8.16 (Forbes-Kelley PRG [37])**.** Let n, δ, γ, r, k be some parameters chosen later. Let $D_1, \ldots, D_r$ be r independent δ-biased distributions and let $T_1, \ldots, T_r$ be r independent γ-almost k-wise independent distributions over $\{0,1\}^n$. We define G_r^{FK} inductively as follows. Let G_1^{FK} be some $320k$-wise independent distribution and let

$$G_{i+1}^{\mathrm{FK}} := D_i + T_i \wedge G_i^{\mathrm{FK}},$$

for all $1 \le i \le r-1$, where $\wedge$ denotes bitwise AND and $+$ denotes bitwise XOR.

Theorem 8.17 (Correctness of Forbes-Kelley PRG [37])**.** *For parameters $n, w \in \mathbb{N}$ and $\varepsilon > 0$, choose the parameters as follows: $r := \lceil \lg n \rceil$, $k := \lceil 3 \lg(nw/\varepsilon) \rceil$, $\gamma := (nw/\varepsilon)^{-9}$, and $\delta := (nw\mathcal{L}/\varepsilon)^{-3}$, where $\mathcal{L} := \sum_{i=1}^{k} \binom{n}{i} w^{1/2}$. Then, $G_r^{\mathrm{FK}} : \{0,1\}^s \to \{0,1\}^n$ is a pseudorandom generator that ε-fools any-order ROBPs of width w, where the seed length s is $O\big(\log(nw/\varepsilon) \cdot \log^2 n\big)$.*

Theorem 8.18. *There exists a $\widetilde{O}\left(\log(nw/\varepsilon)\right) \cdot \mathrm{polylog}(n)$-local pseudorandom generator of seed length $O\left(\log(nw/\varepsilon) \cdot \log^2 n\right)$ that ε-fools any-order n-input ROBPs of width w.*

Proof. We instantiate the Forbes-Kelley pseudorandom generator G_r^{FK} by using the parameters of Theorem 8.17 and using the constructions of generators of Theorem 2.10, Theorem 8.15, and Lemma 2.8 for D_i, T_i, and G_1^{FK}, respectively.

For a distribution $\mathcal{D}$, we denote by $\mathrm{CC}(\mathcal{D})$ the maximum of $\mathrm{CC}(D)$ over all strings D in the range of $\mathcal{D}$. We claim that $\mathrm{CC}\big(G_r^{\mathrm{FK}}\big)$ is at most $\widetilde{O}\left(\log(nw/\varepsilon)\right) \cdot \mathrm{polylog}(n)$.

By Theorem 2.10 and Theorem 8.15, we have $\mathrm{CC}(D_i) \le \widetilde{O}(\log(n/\delta)) \cdot \mathrm{polylog}(n)$ and $\mathrm{CC}(T_i) \le \widetilde{O}(k + \log(n/\gamma)) \cdot \mathrm{polylog}(n)$. Therefore, since $G_{i+1}^{\mathrm{FK}} = D_i + T_i \wedge G_i^{\mathrm{FK}}$, we obtain

$$\mathrm{CC}\big(G_{i+1}^{\mathrm{FK}}\big) \le \mathrm{CC}(D_i) + \mathrm{CC}(T_i) + \mathrm{CC}\big(G_i^{\mathrm{FK}}\big) + O(1)$$

for any $i \in [r-1]$. We also have $\mathrm{CC}\big(G_1^{\mathrm{FK}}\big) \le k \cdot \widetilde{O}(\log n)$ from Lemma 2.8. Therefore,

$$\begin{aligned}\mathrm{CC}\big(G_r^{\mathrm{FK}}\big) &\le r \cdot \widetilde{O}(k + \log(n/\gamma\delta)) \cdot \mathrm{polylog}(n) + k \cdot \widetilde{O}(\log n) \\ &= \widetilde{O}(\log(nw/\varepsilon)) \cdot \mathrm{polylog}(n)\,. \end{aligned}$$

□

Corollary 8.19 (Theorem 8.10, restated)**.** *There exists a* $\widetilde{O}\big((\sqrt{t} + \log(1/\varepsilon)) \cdot \log q\big) \cdot \mathrm{polylog}(n)$*-local pseudorandom generator with seed length* $\widetilde{O}\big((\sqrt{t} + \log(1/\varepsilon)) \cdot \log q\big)$ *that* ε*-fools* q*-state time-*t *n-input DTMs, for any* $t \ge n$*.*

Proof. By Lemma 8.9, any q-state time-t DTM M can be written as the sum of ROBPs $\{P_\alpha\}_{\alpha \in A}$ so that $M(x) = \sum_{\alpha \in A} P_\alpha(x)$, where $|A| \le (tq)^{O(\sqrt{t})}$. We set $\varepsilon' := \varepsilon/|A|$ and use the pseudorandom generator G_r^{FK} of Theorem 8.18 that ε'-fools ROBPs of width w, where $w = (tq)^{O(\sqrt{t})}$. Then, G_r^{FK} is a PRG that ε-fools DTMs because

$$\begin{aligned}\Big|\mathop{\mathbf{Pr}}_z\big[M(G_r^{\mathrm{FK}}(z)) = 1\big] - \mathop{\mathbf{Pr}}_w[M(w) = 1]\Big| &= \Big|\mathop{\mathbf{E}}_z\big[M(G_r^{\mathrm{FK}}(z))\big] - \mathop{\mathbf{E}}_w[M(w)]\Big| \\ &= \Big|\mathop{\mathbf{E}}_{z,w}\big[M(G_r^{\mathrm{FK}}(z)) - M(w)\big]\Big| \\ &\le \sum_{\alpha \in A}\Big|\mathop{\mathbf{E}}_{z,w}\big[P_\alpha(G_r^{\mathrm{FK}}(z)) - P_\alpha(w)\big]\Big| \le \varepsilon. \end{aligned}$$

□

Notes

The results of this chapter are products of joint work with Mahdi Cheraghchi, Shuichi Hirahara, and Yuichi Yoshida [28].

Chapter 9

MKTP Lower Bounds Against Branching Programs

Nečiporuk's method [82] is a standard technique for obtaining branching program lower bounds. Using this method, Nečiporuk proved the first lower bounds against branching programs of almost quadratic size. His argument is based on counting the subfunctions of the function that we want to show is hard. It was an open problem to use this method in obtaining branching program lower bounds for the Minimum KT-complexity Problem (MKTP) which is a resource-bounded version of Kolmogorov complexity.

In this chapter, we use Nečiporuk's method to obtain such a lower bound. In particular, we prove that any branching program of size $o\big(N^2/\log^2 N\big)$ and any non-deterministic (or parity) branching program of size $o(N^{1.5}/\log N)$ cannot compute MKTP on inputs of size N.

In order to apply Nečiporuk's method to MKTP, we need to give a lower bound on the number of distinct subfunctions that can be obtained by fixing all but $O(\log n)$ bits. The idea of counting distinct subfunctions of MKTP is to show that a random restric-

tion which leaves $O(\log n)$ variables free induces different subfunctions with high probability. Specifically, partition the input variables $[n]$ into $m := n/O(\log n)$ blocks, pick $m-1$ strings $\rho := \rho_2 \cdots \rho_m \in (\{0,1\}^{O(\log n)})^{m-1}$ randomly, and consider the restricted function $f|_\rho(\rho_1) := \mathrm{MKTP}(\rho_1\rho_2\cdots\rho_m, \theta)$ for some threshold function θ to be chosen later. Then, the string $\rho_i\rho_2\cdots\rho_m$ is compressible when $i \in \{2,\ldots,m\}$ whereas the string $\rho_1\rho_2\cdots\rho_m$ is not compressible when ρ_1 is chosen randomly. This holds as, in the former case, there exists a $k \in \{2,\ldots,m\}$ such that $\rho_i = \rho_k$ and this yields a description for the string $\rho_i\rho_2\cdots\rho_m$ that is shorter than most of its descriptions in the latter case. Let now θ be an upper bound on the KT complexity of $\rho_i\rho_2\cdots\rho_m$ in the case where $i \in \{2,\ldots,m\}$. Therefore, $f|_\rho(\rho_i) = 1$ for any ρ and $i \in \{2,\ldots,m\}$, and $f|_\rho(\rho_1) = 0$ with high probability over random ρ and ρ_1. This implies that, with high probability over the random restrictions ρ and ρ', it is the case that $f|_\rho \not\equiv f|_{\rho'}$. This is so as, for every $i \in \{2,\ldots,m\}$, the probability over the random restrictions ρ and ρ' that the string ρ_i is such that $f|_{\rho'}(\rho_i) = f|_\rho(\rho_i)$ is small, by the fact that $f|_\rho(\rho_i) = 1$ for any ρ and the fact that $f|_{\rho'}(\rho_i) = 0$ with high probability over random ρ_i and ρ' (and therefore with high probability over random ρ and ρ' as well).

Unfortunately, the probability that $f|_\rho \equiv f|_{\rho'}$ holds may not be exponentially small. As a consequence, a lower bound on the number of distinct subfunctions that can be directly obtained from this fact may not be exponential. In contrast, we need to prove an exponential lower bound on the number of distinct subfunctions in order to obtain a strong lower bound via Nečiporuk's method.

In order to make the argument work, we exploit the symmetry of information for (resource-unbounded) Kolmogorov complexity and Kolmogorov randomness. Instead of picking ρ and ρ' randomly, we keep a set P which contains restrictions ρ that induce distinct subfunctions. Starting from $P := \emptyset$, we add one Kolmogorov-random restriction ρ to P so that the property

of P is preserved. By using the symmetry of information for Kolmogorov complexity, we can argue that one can add a restriction to P until P becomes as large as $2^{\Omega(n)}$, which proves that the number of distinct subfunctions of MKTP is exponentially large.

In Section 9.1, we give some background information regarding Kolmogorov complexity and MKTP. In Section 9.2, we explain Nečiporuk's method for obtaining branching program lower bounds and, in Section 9.3, we prove Theorem 1.13.

9.1 Preliminaries

9.1.1 Non-deterministic and Parity Branching Programs

Here, we define various types of branching programs (see Section 2.1.3). A branching program is called *non-deterministic* if some of its vertices have an arbitrary number of outgoing edges (i.e., if this number is not 2) or if some of its vertices have edges that do not refer to the same input variable. Non-deterministic branching programs may also have *unlabelled* edges, as well. Due to the nature of a non-deterministic branching program, it is possible that a computation never reaches either h_0 or h_1 as there can be some node with edges that their labels are all false according to the input at hand; in this case, we assume that the computation halts in a rejecting state.

A *non-deterministic branching program computes a function* $f : \{0,1\}^n \to \{0,1\}$ if for every $x \in \{0,1\}^n$ such that $f(x) = 1$ there is some s-h_1 path, and for every $x \in \{0,1\}^n$ such that $f(x) = 0$ all computations end in a rejecting state.

A *parity branching program* is a branching program that has counting semantics. That is, a parity branching program computes a function $f : \{0,1\}^n \to \{0,1\}$ if for every $x \in \{0,1\}^n$ such that $f(x) = 1$ there is an odd number of s-h_1 paths, and for every $x \in \{0,1\}^n$ such that

$f(x) = 0$ there is an even number of s-h_1 paths.

We define the *size* of a non-deterministic or parity branching program to be the number of its labelled edges.

9.1.2 Symmetry of Information

We will make use of the symmetry of information of (resource-unbounded) Kolmogorov complexity.

Lemma 9.1. *There exists a constant $c > 0$ such that, for any strings $x, y \in \{0,1\}^*$,*

$$\mathrm{K}(xy) \geq \mathrm{K}(x) + \mathrm{K}(y \mid x) - c \log \mathrm{K}(xy)\,.$$

9.1.3 The Minimum KT-complexity Problem

We require the following definition of (parameterized) MKTP.

Definition 9.2. Let $c > 0$ be as in Lemma 2.11. Let $\theta : \mathbb{N} \to \mathbb{N}$ be a function. The *Minimum KT-complexity Problem of parameter θ (*MKTP$[\theta]$*)* is defined as follows.

- Input: A string $x \in \{0,1\}^*$.
- Question: Is there a program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ such that

  $$U^{\Pi}\big(i, 1^t\big) = x_i$$

  for all $1 \leq i \leq |x|$, and $|\Pi| + t \leq \theta(|x|)$?

9.2 Nečiporuk's Method for Branching Program Lower Bounds

For a function $f : \{0,1\}^n \to \{0,1\}$, we partition the input variables $[n]$ into disjoint blocks $V_1, \ldots, V_m$, where $|V_i| = v$ for each $i \in [m]$ and $n = vm$. (The parameter $v = O(\log n)$ will be

chosen later.) The idea of the Nečiporuk's method is to lower-bound the number of subfunctions that arise when one restricts variables from V_i, for some $1 \leq i \leq m$. For each $i \in [m]$, we define $c_i(f)$ to be the number of distinct functions $f|_\rho$ such that $\rho : [n] \to \{0, 1, *\}$ is a restriction with $\rho^{-1}(*) = V_i$.

The Nečiporuk method can be then summarized as follows.

Theorem 9.3 (Nečiporuk [82]; cf. [60, Theorem 15.1])**.** *The size of a branching program computing f is at least $\Omega\left(\sum_{i=1}^{m} \log c_i(f) / \log\log c_i(f)\right)$. The size of a non-deterministic branching program or a parity branching program computing f is at least $\Omega\left(\sum_{i=1}^{m} \sqrt{\log c_i(f)}\right)$.*

9.3 MKTP Lower Bounds Against Branching Programs

In this section, we develop a proof technique for applying Nečiporuk's method to MKTP and prove Theorem 1.13.

Our main technical result of this section is the following.

Theorem 9.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be $\mathrm{MKTP}[\theta]$ on n-bit inputs for $\theta := n - 3c\log n - 4$, where $c > 0$ is a universal constant. Then, for every $i \in [m]$, it holds that $c_i(f) = 2^{\Omega(n)}$.*

The lower bounds for branching programs (Theorem 1.13) immediately follow from Theorem 9.4 and Theorem 9.3.

In our proof of Theorem 9.4, we only need the following two properties of KT complexity.

1. The resource-unbounded Kolmogorov complexity provides a lower bound on the KT-complexity. That is, $\mathrm{K}(x) \leq \mathrm{KT}(x)$ for any $x \in \{0,1\}^*$.

2. For any strings $\rho_1, \ldots, \rho_m \in \{0,1\}^v$ such that there exist distinct indices $i \neq j \in [m]$ such that $\rho_i = \rho_j$, we have $\mathrm{KT}(\rho_1 \cdots \rho_m) \leq (m-1) \cdot v + O(\log n)$. This is because each

bit of the string $\rho_1 \cdots \rho_m$ can be described by the strings $\{\rho_1, \ldots, \rho_m\} \setminus \{\rho_j\}$ and the index $j \in [m]$ in time $O(\log n)$.

For simplicity, we focus on the case when $i = 1$; the other case can be proved similarly. The idea of the proof is the following. Imagine that we pick $\rho \in \{*\}^{V_1} \times \{0,1\}^{V_2 \cup \cdots \cup V_m}$ uniformly at random. (Here we identify a restriction with a string in $\{0,1,*\}^{[n]}$.) We denote by $\rho_2 \in \{0,1\}^{V_2}, \ldots, \rho_m \in \{0,1\}^{V_m}$ the random bits such that $\rho = *^{V_1}\rho_2 \cdots \rho_m$. We will sometimes identify $\rho_2 \cdots \rho_m$ with ρ.

Consider the function $f|_\rho : \{0,1\}^{V_1} \to \{0,1\}$ obtained by restricting f by ρ. Then, we expect that $f|_\rho(\rho_i) = 1$ for any $i \in \{2, \ldots, m\}$ since $\mathrm{KT}(\rho_i \rho_2 \cdots \rho_m)$ is small, whereas $f|_\rho(U) = 0$ for a random $U \sim \{0,1\}^{V_1}$ with high probability. Thus, the function $f|_\rho$ is likely to be distinct for a randomly chosen ρ.

In order to make the argument formal, we proceed as follows. Pick ρ randomly. Then we add it to a set P while keeping the promise that the map $\rho \in P \mapsto f|_\rho$ is injective. We will show that one can keep adding ρ until the size of P becomes exponentially large.

We focus on restrictions ρ such that ρ is Kolmogorov-random. To this end, define

$$R := \left\{ \rho \in \{0,1\}^{V_2 \cup \cdots \cup V_m} \mid \mathrm{K}(\rho) \geq |\rho| - 1 \right\}$$

as the set of Kolmogorov-random restrictions ρ. By the standard counting argument, we have

$$\Pr_\rho[\rho \notin R] \leq \sum_{i=1}^{|\rho|-2} \frac{2^i}{2^{|\rho|}} \leq \frac{1}{2}.$$

The following lemma is the key for counting the number of distinct subfunctions.

Lemma 9.5. *Let $\rho' \in R$ be an arbitrary restriction. If $f|_\rho \equiv f|_{\rho'}$, then $\mathrm{K}(\rho_i \mid \rho') \leq 2c \log n + 1$ for any $i \in \{2, \ldots, m\}$.*

Proof. Let $\theta := n - v + c \log n$. For each $i \in [m] \setminus \{1\}$,

$$\mathrm{KT}(\rho_i \rho_2 \cdots \rho_m) \leq |\rho_2| + \cdots + |\rho_m| + O(\log n) \leq (m-1) \cdot v + c \log n \leq \theta.$$

This means that $\rho_i \rho_2 \cdots \rho_m$ is a YES instance of MKTP$[\theta]$. Therefore, we have $1 = f|_\rho(\rho_i) = f|_{\rho'}(\rho_i)$, which implies that $\mathrm{KT}(\rho_i \rho'_2 \cdots \rho'_m) \leq \theta$. By the symmetry of information,

$$\theta \geq \mathrm{KT}\big(\rho_i \rho'_2 \cdots \rho'_m\big) \geq \mathrm{K}\big(\rho_i \rho'_2 \cdots \rho'_m\big) \geq \mathrm{K}\big(\rho'_2 \cdots \rho'_m\big) + \mathrm{K}\big(\rho_i \mid \rho'_2 \cdots \rho'_m\big) - c \log n.$$

Since $\rho' \in R$, we have $\mathrm{K}(\rho'_2 \cdots \rho'_m) \geq v(m-1) - 1 = n - v - 1$. Therefore,

$$\mathrm{K}\big(\rho_i \mid \rho'_2 \cdots \rho'_m\big) \leq \theta + c \log n - (n - v - 1) = 2c \log n + 1. \qquad \square$$

Now we set $v := 4c \log n + 4$. Then, for any $\rho' \in R$,

$$\begin{aligned}
\Pr_\rho[f|_\rho \equiv f|_{\rho'}] &\leq \Pr[\forall i \in [m] \setminus \{1\},\ \mathrm{K}\big(\rho_i \mid \rho'\big) \leq v/2 - 1] \\
&\leq (2^{v/2}/2^v)^{m-1} \\
&= 2^{-n/2+v/2} \\
&\leq 2^{-n/3}.
\end{aligned}$$

In particular, for any $P \subseteq R$, by the union bound, we obtain

$$\Pr_\rho[\exists \rho' \in P,\ f|_\rho \equiv f|_{\rho'}] \leq |P| \cdot 2^{-n/3}.$$

Therefore,

$$\Pr_\rho[\rho \notin R \text{ or } \exists \rho' \in P,\ f|_\rho \equiv f|_{\rho'}] \leq 1/2 + |P| \cdot 2^{-n/3},$$

which is strictly less than 1 if $|P| < 2^{n/3-1}$. To summarize, we established the following property.

Corollary 9.6. *For any $P \subseteq R$ such that $|P| < 2^{n/3-1}$, there exists a restriction ρ such that $\rho \in R$ and $f|_\rho \not\equiv f|_{\rho'}$ for any $\rho' \in P$.*

In light of this, we can construct a large set P such that the map $\rho \in P \mapsto f|_\rho$ is injective as follows: Starting from $P := \emptyset$, add a restriction $\rho \in R$ such that $f|_\rho \not\equiv f|_{\rho'}$ for any $\rho' \in P$, whose existence is guaranteed by Corollary 9.6 if $|P| < 2^{n/3-1}$. In this way, we obtain a set P such that $|P| \geq 2^{n/3-1}$ and each $f|_\rho$ is distinct for any $\rho \in P$. We conclude that $c_1(f) \geq |P| \geq 2^{n/3-1}$. This completes the proof of Theorem 9.4.

Notes

The results of this chapter are products of joint work with Mahdi Cheraghchi, Shuichi Hirahara, and Yuichi Yoshida [28].

Chapter 10

Connections to Cryptography

One-way functions (OWFs) are fundamental cryptographic primitives. For many years it was an open problem to demonstrate an NP-complete problem whose average-case hardness is *equivalent* to the existence of OWFs. In this chapter, we make progress in this problem and we show that there is an NP-complete problem whose average-case hardness is *almost* equivalent to the existence of OWFs.

The problem that we are considering is a conditional variant of the Minimum KT-complexity Problem (MKTP), which we call McKTP. We show the following. First, we prove that if McKTP is average-case hard on a polynomial fraction of its instances, over the uniform distribution, then there exist OWFs (Theorem 1.15). Then, we observe that McKTP is NP-complete under polynomial-time randomized reductions (Theorem 1.16). Finally, we prove that the existence of OWFs implies the nontrivial average-case hardness of McKTP (Theorem 1.17), over the uniform distribution. Thus the existence of OWFs is inextricably linked to the average-case hardness of this NP-complete problem.

We now provide some intuition regarding the proofs of Theorem 1.15, Theorem 1.16, and

Theorem 1.17.

1. Theorem 1.15 is proved by observing that a recent result by Liu and Pass [70], whereby they prove that the average-case hardness of time-bounded Kolmogorov complexity K^t yields OWFs, can be adjusted to the case of McKTP as well (see Theorem 10.28).

   The three properties of time-bounded Kolmogorov complexity K^t, for some $t : \mathbb{N} \to \mathbb{N}$ where $t(n) \geq n$ for all $n \in \mathbb{N}$, that are used by Liu and Pass, are as follows.

   (a) One can create a string of low time-bounded Kolmogorov complexity in polynomial time. This can be done by running a universal Turing machine U on some string, for polynomially-many steps, and subsequently recording the output of U.

   (b) For any string x, the possible values of its K^t complexity are polynomially-many in $|x|$. In fact, there is a $c > 0$ such that, for any function $t : \mathbb{N} \to \mathbb{N}$ such that $t(n) \geq n$ for all $n \in \mathbb{N}$, and any string x, the possible values of $K^t(x)$ are at most $|x| + c$.

   (c) The following domination property holds. Let $x^* \in \{0,1\}^n$ be a string, and $c > 0$ be as in Item 1b. Then,

$$\mathop{\mathbf{Pr}}_{\Pi \sim \{0,1\}^{n+c}} \left[U\left(\Pi, 1^{t(n)}\right) = x^* \right] \geq \frac{1}{2^{n+c}} = \frac{2^{-n}}{2^c} \geq \frac{\mathbf{Pr}_{x \sim \{0,1\}^n}[x = x^*]}{\mathrm{poly}(n)}.$$

   As it turns out, all of these properties are satisfied even when one considers McKTP. This is true for the following reasons.

   (d) One can create a string of low conditional KT complexity in polynomial time. As above, this can be done by running a universal *oracle* Turing machine U, that has oracle access to some string y, on a string x for $t \leq \mathrm{poly}(|x|)$ steps, and subsequently recording the output of U^y.

(e) For any string x, the possible values of its KT complexity given any string y are polynomially-many in $|x|$. In fact, there is a $c > 0$ such that for any string x, the possible values of the conditional KT complexity of (x, y) are at most $|x| + c \log |x|$.

(f) The following domination property holds (for KT *and* conditional KT complexity). Let $x^* \in \{0,1\}^n$ be a string, $c > 0$ be as in Item 1e, and $n_c := n + c \log n$. Then, for all $y \in \{0,1\}^*$, it is the case that

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\substack{\Pi \sim \{0,1\}^{n_c},\ t \in [n_c]: \\ |\Pi| + t \leq n_c}} \left[\text{for all } 1 \leq i \leq n,\ U^{\Pi, y}\left(i, 1^t\right) = x^*_i \right] &\geq \frac{1}{2^{n_c + \log n_c}} \\ &= \frac{2^{-n}}{n^c \left(n + c \log n\right)} \\ &= \frac{\mathbf{Pr}_{x \sim \{0,1\}^n}[x = x^*]}{\text{poly}(n)}. \end{aligned}$$

2. Theorem 1.16 is proved by

   (a) noting that McKTP is in NP (see Lemma 10.13) and

   (b) showing the NP-hardness of McKTP (see Corollary 10.43). This is done by giving a polynomial-time randomized reduction from Set Cover, which is NP-hard to approximate (see Corollary 10.42), to an appropriate gap version of McKTP; see Corollary 10.41. Note that this step closely mimics the proof of Ilango [52] for the NP-hardness of Minimum Oracle Circuit Size Problem (MOCSP).

3. Theorem 1.17 is proved by giving a proof of its contrapositive statement, as explained by the items below.

   (a) Assume that McKTP is easy on average under the uniform distribution.

   (b) By a corollary of Ilango, Loff, and Oliveira (see Lemma 10.23), for all $a \geq 1$, there exists a learning algorithm for $\mathsf{SIZE}[n^a]$ that works for infinitely many $n \in \mathbb{N}$; see

Lemma 10.31.

(c) By a learner-to-distinguisher reduction (see Lemma 10.34), for every polynomial-time computable Boolean function family $\{f_y\}_{y\in\{0,1\}^*}$, there is a distinguisher for $\{f_y\}_{y\in\{0,1\}^*}$.

(d) By the correctness of the works by Håstad, Implagliazzo, Levin, and Luby [46], and Goldreich, Goldwasser, and Micali [39], there are no OWFs.

In Section 10.1 we give some background knowledge and useful facts. We prove Theorem 1.15 in Section 10.2 and Theorem 1.17 in Section 10.3. Finally, we prove Theorem 1.16 in Section 10.4.

10.1 Preliminaries

10.1.1 Probability Theory

We will use the following useful facts from probability theory.

Lemma 10.1 (Averaging argument)**.** *If $X \in [0,1]$ is a random variable with $\mu := \mathbf{E}[X]$, then for all $0 < c < 1$ it is the case that*

$$\mathbf{Pr}[X \geq c\mu] \geq (1-c)\,\mu.$$

Lemma 10.2 (Markov's inequality)**.** *If X is a non-negative random variable with $\mu := \mathbf{E}[X]$, then for all $k > 0$ it is the case that*

$$\mathbf{Pr}[X \geq k\mu] \leq \frac{1}{k}.$$

Lemma 10.3 (Chernoff bound)**.** *Let $n \in \mathbb{N}$ and $X_1, \ldots, X_n$ be Boolean random variables that are independent and identically distributed. Let $X := \sum_{i=1}^{n} X_i$ and $\mu := \mathbf{E}[X]$. Then, for all*

$0 \leq \delta \leq 1$, *it is the case that*

$$\mathbf{Pr}[X \geq (1+\delta)\,\mu] \leq e^{-\frac{\delta^2 \mu}{3}}.$$

10.1.2 Circuit Complexity

We require the following trivial lower bound on the circuit complexity of an arbitrary Boolean function.

Lemma 10.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function. Then,* $\mathrm{CC}(f) \geq n$.

The following lemma will be useful for us.

Lemma 10.5. *Let $s : \mathbb{N} \to \mathbb{N}$ be a size function. Then, there exists an algorithm that, on input a description $d_C \in \{0,1\}^*$ of a circuit C that has $n \in \mathbb{N}$ inputs and size $s(n)$, whereby $|d_C| \leq O(s(n) \log s(n))$ (as in Lemma 2.1), and a string $x \in \{0,1\}^n$, outputs $C(x) \in \{0,1\}$ in time $O\Big(s(n)^2 \log s(n)\Big)$.*

Proof. Let C be a circuit that has n inputs and size $s(n)$, and let $x \in \{0,1\}^n$ be a string. Then, on input x, any gate of C can be evaluated in a bottom-up fashion by parsing the description $d_C \in \{0,1\}^*$ of C a constant number of times. The idea described above can be implemented as an algorithm that runs in time

$$s(n) \cdot O(|d_C|) \leq s(n) \cdot O(s(n) \log s(n)) = O\Big(s(n)^2 \log s(n)\Big). \qquad \square$$

10.1.3 Uniform Algorithms

We say that an algorithm A is a *PPT algorithm* if A is a probabilistic polynomial-time algorithm. If A is a PPT algorithm that runs in time $p(n)$ for a polynomial p, then we denote by $A(x, r)$ the output of A on input $x \in \{0,1\}^*$ using random bits $r \in \{0,1\}^{p(|x|)}$. We say that an

algorithm A is a *PPT oracle algorithm* if A is a PPT algorithm that has access to some oracle. If A is a PPT oracle algorithm that runs in time $p(n)$ for a polynomial p and has access to an oracle for a language $L \subseteq \{0,1\}^*$, then we denote by $A^L(x,r)$ the output of A^L on input $x \in \{0,1\}^*$ using random bits $r \in \{0,1\}^{p(|x|)}$.

We require the following observation, which upper bounds the circuit complexity of uniform computations.

Lemma 10.6 ([91])**.** *Let $T : \mathbb{N} \to \mathbb{N}$ be a function. Let $f : \{0,1\}^n \to \{0,1\}$ be a function computable by a Turing machine in time $O(T(n))$. Then, there exists a circuit of size $O(T(n) \log T(n))$ that computes f.*

10.1.4 KT complexity, Revisited

Another Universal Turing Machine

In what follows, we fix some UTM U. Let $y, \Pi, z \in \{0,1\}^*$ and $t \in \mathbb{N}$. The notation $U^{\Pi,y}\big(z, 1^t\big)$ denotes the output of U when U runs the program Π on input z for at most t steps, given that U has extended oracle access to program Π and standard oracle access to auxiliary string y. These oracle access notions are defined as follows.

1. *Standard oracle access to auxiliary string y* means that U has a standard oracle tape T_y of $\log |y|$ cells, and that in order to read a bit y_i of y, whereby $1 \leq i \leq |y|$, the machine U has to write $i \in \{0,1\}^{\log |y|}$ on T_y and then enter a question state. In the next step, the contents of T_y are erased and replaced by a bit b such that $b = y_i$.

   One important aspect of our choice of U is that, for every auxiliary string $y \in \{0,1\}^*$ and $1 \leq i \leq \log |y|$, the oracle query $y_i \stackrel{?}{=} 1$ is such that *requires* time $\log |y|$, and can be implemented in time $O(\log |y|)$.

2. *Extended oracle access to program* Π means that U has a tape T_Π of $|\Pi|$ cells that contain Π, and the head of T_Π has *both* the ability to jump to an indexed location $1 \leq i \leq |\Pi|$ of T_Π, namely $T_\Pi[i] = \Pi_i$, *and* to move left and right on T_Π. Note that in the former case the index i is written in a separate tape of $\log |\Pi|$ cells, specifically allocated for that purpose. (So extended oracle access implies the existence of two tapes that help facilitate the oracle query.)

The notation $U^{\Pi,y}(z)$ denotes the output of U when U runs the program Π on input z, until Π halts (if this is the case, otherwise Π runs forever), whereby U has extended oracle access to Π and standard oracle access to y.

Remark 10.7. In this chapter, we will assume that whenever U is given oracle access to a program Π, this access will be *extended*, and whenever U is given oracle access to an auxiliary string y, this access will be *standard*. This is mainly to avoid unnecessary complications in the proof of Theorem 1.16, and in particular in the proof of Lemma 10.38 where it is convenient to have sequential access to Π (see the proof of Claim 10.39), while requiring that each query to y uses logarithmic time. Moreover, our choice also maintains the trivial upper bound on KT complexity (see Lemma 2.11), which requires oracle access to Π. This will be useful in our analyses, and especially in the proof of Theorem 1.15 (see the proof of Theorem 10.28).

Finally, we will assume that the output of U will either be 1 or 0, on any input.

Some Properties of KT Complexity

By Lemma 2.11, we get the following corollary.

Corollary 10.8. *There is a* $c > 0$ *such that for all* $x, y \in \{0,1\}^*$ *it is the case that* $\mathrm{KT}(x \mid y)$ *is at most* $|x| + c \log |x|$.

Lemma 10.9. *Let $n, m \in \mathbb{N}$ be such that $n \leq m$, let $\sigma > 0$, and let $s := n^{1/\sigma}$. Then, for all $y \in \{0,1\}^m$ there exists $x \in \{0,1\}^n$ such that $\mathrm{KT}(x \mid y) \leq s$.*

Proof. This can be seen by setting $x := y_1 \cdots y_n \in \{0,1\}^n$. In this case, $\mathrm{KT}(x \mid y) \leq c \log n \leq n^{1/\sigma} = s$, where c is from Corollary 10.8. □

Lemma 10.10. *Let $n \in \mathbb{N}$ be sufficiently large, $m \in \mathbb{N}$, and $c > 1$. Then, it is the case that*

$$\mathop{\mathbf{Pr}}_{\substack{x \sim \{0,1\}^n, \\ y \sim \{0,1\}^m}} \Big[\mathrm{KT}(x \mid y) \leq n^{1/c} \Big] = o(1) \,,$$

where x and y are independent and uniformly random.

Proof. We have that

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{x,y} \Big[\mathrm{KT}(x \mid y) \leq n^{1/c} \Big] &= \sum_y \mathop{\mathbf{Pr}}_{x} \Big[\mathrm{KT}(x \mid y) \leq n^{1/c} \Big] \cdot \frac{1}{2^m} \\
&= \sum_y \frac{\big| \big\{ x \in \{0,1\}^n \mid \mathrm{KT}(x \mid y) \leq n^{1/c} \big\} \big|}{2^n} \cdot \frac{1}{2^m} \\
&\leq \sum_y \frac{\big| \big\{ \Pi \in \{0,1\}^* \mid |\Pi| \leq n^{1/c} \big\} \big|}{2^n} \cdot \frac{1}{2^m} ,
\end{aligned}$$

since $\mathrm{KT}(x \mid y) \leq n^{1/c}$ implies that there exists a program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ such that $U^{\Pi, y}\big(i, 1^t\big) = x_i$ for all $1 \leq i \leq n$, and $|\Pi| < |\Pi| + t \leq n^{1/c}$, or

$$\begin{aligned}
&= \frac{\big| \big\{ \Pi \in \{0,1\}^* \mid |\Pi| \leq n^{1/c} \big\} \big|}{2^n} \\
&\leq \frac{2^{n^{1/c}+1} - 1}{2^n} \\
&\leq \frac{2^{n^{1/c}+1}}{2^n} \\
&= o(1) \,.
\end{aligned}$$

□

Lemma 10.11. *For all $f : \{0,1\}^n \to \{0,1\}$, $1 \leq t \leq 2^n$, and $x_1, \ldots, x_t \in \{0,1\}^n$, it is the case that*

$$\mathrm{KT}((f(x_1), \ldots, f(x_t)) \mid (x_1, \ldots, x_t)) \leq O\Big(\mathrm{CC}(f)^3 \Big) \,.$$

Proof. Let C_f be a minimum-size circuit that computes f. Let Π be a program as follows:

On input $1 \leq i \leq n$, compute and output $C_f(x_i)$.

By Lemma 2.1, we can assume that the description of C_f has size $O(\mathrm{CC}(f) \log \mathrm{CC}(f))$, so we have that $|\Pi| \leq O(\mathrm{CC}(f) \log \mathrm{CC}(f))$. By Lemma 10.5 and Lemma 10.4, the running time of Π is at most

$$O(\log n) + O(n \log n) + O\Big(\mathrm{CC}(f)^2 \log \mathrm{CC}(f)\Big) \leq O\Big(\mathrm{CC}(f)^3\Big).$$

Therefore, $\mathrm{KT}((f(x_1), \ldots, f(x_t)) \mid (x_1, \ldots, x_t)) \leq O\Big(\mathrm{CC}(f)^3\Big)$. □

10.1.5 Minimum Conditional KT-complexity Problem and Variants

We give here formal definitions of the computational problems that we will consider in this chapter. These are the decision and search variants of McKTP.

Definition 10.12 (Decision variant)**.** Let $c > 0$ be as in Corollary 10.8. Let $n \in \mathbb{N}$ and $m : \mathbb{N} \to \mathbb{N}$. The *Minimum m-Conditional KT-complexity Problem of dimension n (*$\mathrm{McKT}^m\mathrm{P}$ *of dimension n)* is defined as follows.

- Input: Strings $x \in \{0,1\}^n$, $y \in \{0,1\}^{m(n)}$, and a parameter $0 \leq \theta \leq n + c \log n$ in binary.
- Question: Is there a program $\Pi \in \{0,1\}^*$ and a run-time bound $t \in \mathbb{N}$ such that

$$U^{\Pi,y}\big(i, 1^t\big) = x_i$$

for all $1 \leq i \leq n$, and $|\Pi| + t \leq \theta$?

The following result is a standard observation.

Lemma 10.13. *For all polynomial-time computable functions $m : \mathbb{N} \to \mathbb{N}$, it is the case that* $\mathrm{McKT}^m\mathrm{P}$ *of dimension n is in* NP.

10.1.6 Pseudorandom Generators That Fool PPT Algorithms

We recount the notion of fooling a PPT algorithm.

Definition 10.14. Let $\ell : \mathbb{N} \to \mathbb{N}$ be such that $\ell(n) < n$ for all $n \in \mathbb{N}$, and $G = \left\{G_n : \{0,1\}^{\ell(n)} \to \{0,1\}^n\right\}_{n\in\mathbb{N}}$ be a function. Let A be a PPT algorithm that on inputs of size $n \in \mathbb{N}$ runs in time $q(n)$ for some polynomial q. Let also $\varepsilon : \mathbb{N} \to (0,1)$. We say that G *is a function that ε-fools A* if for all $n \in \mathbb{N}$

$$\left| \Pr_{\substack{x\sim\{0,1\}^n,\\ r\sim\{0,1\}^{q(n)}}} [A(x,r) = 1] - \Pr_{\substack{y\sim\{0,1\}^{\ell(n)},\\ r\sim\{0,1\}^{q(n)}}} [A(G(y),r) = 1] \right| \leq \varepsilon(n)\,.$$

10.1.7 One-way Functions

In the following, a function μ is said to be *negligible* if for every polynomial p there exists a $n_0 \in \mathbb{N}$ such that for all naturals $n > n_0$ it is the case that $\mu(n) \leq 1/p(n)$.

Definition 10.15. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. We say that f is a *one-way function (OWF)* if for every PPT algorithm A there exists a negligible function μ such that for all $n \in \mathbb{N}$ it is the case that

$$\Pr_{x\sim\{0,1\}^n,r}\left[A(1^n, f(x), r) \in f^{-1}(f(x))\right] < \mu(n)$$

where the size of r is equal to the running time of A.

Håstad, Impagliazzo, Levin, Luby [46] have shown that the existence of OWFs implies the existence of PRGs.

Theorem 10.16 (OWFs imply PRGs; see Håstad, Impagliazzo, Levin, Luby [46])**.** *If there exists a OWF, then for every $c > 0$ there exists a function $G = \{G_n\}_{n\in\mathbb{N}}$, whereby G_n :*

$\{0,1\}^{n^{1/c}} \to \{0,1\}^n$ *for all* $n \in \mathbb{N}$*, such that for every PPT algorithm* A *there is a negligible function* $\varepsilon : \mathbb{N} \to (0,1)$ *such that* G *is a function that* ε*-fools* A*.*

We will also employ the following weaker notion of OWFs.

Definition 10.17. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time computable function. We say that f is an *α-weak one-way function (α-weak OWF)* if for every PPT algorithm A and all sufficiently large $n \in \mathbb{N}$ it is the case that

$$\Pr_{x \sim \{0,1\}^n, r}\left[A(1^n, f(x), r) \in f^{-1}(f(x))\right] < 1 - \alpha(n)$$

where the size of r is equal to the running time of A. We say that f is a *weak one-way function (weak OWF)* if there exists some polynomial $q > 0$ such that f is a $(1/q)$-weak OWF.

Yao [117] proved that the existence of weak OWFs implies the existence of OWFs.

Theorem 10.18 ([117])**.** *Assume that there exists a weak one-way function. Then there exists a one-way function.*

10.1.8 Average-case Hardness/Easiness

Decision Problems

A *heuristic* H is a PPT algorithm that, on input any $x \in \{0,1\}^n$, outputs a value in $\{0,1\}$ along each computation path.

Definition 10.19 (Average-case hardness)**.** Let $\alpha : \mathbb{N} \to [0,1]$ be a failure parameter function. We say that a function $f : \{0,1\}^n \to \{0,1\}$ is *α-hard-on-average (α-HoA)* if for all heuristics H and all sufficiently large $n \in \mathbb{N}$ it is the case that

$$\Pr_{x \sim \{0,1\}^n, r}[H(x,r) = f(x)] \leq 1 - \alpha(n)$$

where the size of r is equal to the running time of H.

Definition 10.20 (Average-case easiness)**.** Let $\alpha : \mathbb{N} \to [0,1]$ be a success parameter function. We say that a function $f : \{0,1\}^n \to \{0,1\}$ is *α-easy-on-average (α-EoA)* if f is not $(1-\alpha)$-hard-on-average; that is, if there exists some heuristic H such that for infinitely many $n \in \mathbb{N}$ it is the case that

$$\Pr_{x \sim \{0,1\}^n, r}[H(x,r) = f(x)] > \alpha(n)$$

where the size of r is equal to the running time of H.

Infinitely Often Average-case Easiness of McKTP, With Advantage

We require the following definition, which is inspired by Ilango, Loff, and Oliveira [53].

Definition 10.21 (Following Ilango, Loff, and Oliveira [53])**.** Let $m : \mathbb{N} \to \mathbb{N}$, $s : \mathbb{N} \to \mathbb{N}$ be such that $s(n) < n/4$ for all $n \in \mathbb{N}$, and $\alpha : \mathbb{N} \to [0,1]$. A probabilistic algorithm B *solves* McKT^{m}P *of dimension n with advantage α for a size parameter s and infinitely many $n \in \mathbb{N}$* if, for infinitely many $n \in \mathbb{N}$, it is the case that

$$\left| \Pr_{y,r}[B(1^n, x, y, r) = 1] - \Pr_{z,y,r}[B(1^n, z, y, r) = 1] \right| \geq \alpha(n)$$

where $y \in \{0,1\}^{m(n)}$ and $z \in \{0,1\}^n$ are uniformly random, $x = x(y) \in \{0,1\}^n$ is arbitrary and such that $\mathrm{KT}(x \mid y) \leq s$ (which exists by Lemma 10.9), and $r \in \{0,1\}^*$ has size equal to the running time of B. In this case, we may also say that McKT^{m}P of dimension n is *easy-on-average with advantage α for a size parameter s and infinitely many $n \in \mathbb{N}$ (EoA with advantage α for a size parameter s and infinitely many $n \in \mathbb{N}$).*

10.1.9 Learning Algorithms

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, an *example oracle for f*, denoted $\mathrm{EX}(f)$, is a procedure that when invoked returns a pair $(x, f(x))$ where $x \sim \{0,1\}^n$.

Let $0 < \varepsilon < 1$. We say that a circuit C with n inputs is *ε-close* to a function $f : \{0,1\}^n \to \{0,1\}$ if

$$\Pr_{x \sim \{0,1\}^n}[C(x) \neq f(x)] \leq \varepsilon.$$

We follow the intuitive notion of learning by Valiant [111].

Definition 10.22 (Infinitely often learnability; following Valiant [111])**.** A probabilistic algorithm *infinitely often learns* a class of Boolean functions $\mathcal{F}$ with accuracy error ε and confidence error δ if, for all $f \in \mathcal{F}$ of the form $f = \{f_n\}_{n \in \mathbb{N}}$, whereby $f_n : \{0,1\}^n \to \{0,1\}$ for all $n \in \mathbb{N}$, and infinitely many $n \in \mathbb{N}$, it is the case that

$$\Pr_{\mathrm{EX}(f_n), r}\left[A^{\mathrm{EX}(f_n)}(1^n, r) \text{ outputs a circuit that is } \varepsilon(n)\text{-close to } f_n\right] \geq 1 - \delta(n)\,,$$

where the size of r is equal to the running time of A.

We will use the following result, which is inspired by Ilango, Loff, and Oliveira [53].

Lemma 10.23 (Following Ilango, Loff, and Oliveira [53, Theorem 36, Item (2)])**.** *If there exists $c > 1$ such that for all $0 < \gamma < 1$ McKT^{m}P of dimension t and $m := t^{1+\gamma}$ can be solved on average with advantage $1/10$ for a size parameter $s := t^{1/c}$ and infinitely many $t \in \mathbb{N}$, then for every $a \geq 1$ the class $\mathsf{SIZE}[n^a]$ can be infinitely often learned in polynomial time with accuracy error $1/n$ and confidence error $1/n$.*

Proof. Let $a \geq 1$ be arbitrary, let $c > 1$ be as in the statement of Lemma 10.23, let $\gamma := 1/(c \cdot 4a)$, and let B be the PPT algorithm that witnesses the fact that McKT^{m}P of dimension t and $m := t^{1+\gamma}$ can be solved on average with advantage $1/10$ for a size parameter $s := t^{1/c}$ and infinitely many $t \in \mathbb{N}$. Let $t \in \mathbb{N}$ be sufficiently large and such that B satisfies its average-case guarantees on inputs of dimension $n := t^\gamma$. Let $N := t + m(t) = t + t^{1+\gamma} = t + tn$ be the size of the instances of McKT^{m}P of dimension t. Let q be a polynomial such that B runs in time

$q(N)$ on inputs of size N. We will prove that $\mathsf{SIZE}[n^a]$ can be learned in polynomial time with accuracy error $1/n$ and confidence error $1/n$.

By examining the work of Ilango, Loff, and Oliveira [53, Theorem 36, Item (2)], it would suffice to show that there exists a PPT algorithm B' such that for infinitely many $n \in \mathbb{N}$ and all $f \in \mathsf{SIZE}[n^a]$ it is the case that

$$\left| \Pr_{\{x_i\}_{i=1}^t, r} \left[B'(1^n, (x_1, f(x_1)), \ldots, (x_t, f(x_t)), r) = 1 \right] - \Pr_{\{x_i\}_{i=1}^t, b, r} \left[B'(1^n, (x_1, b_1), \ldots, (x_t, b_t), r) = 1 \right] \right| \geq \frac{1}{10} \tag{10.1}$$

whereby $x_1, \ldots, x_t \in \{0,1\}^n$, $b \in \{0,1\}^t$, and $r \in \{0,1\}^{\mathrm{poly}N}$ are independent and uniformly random. The reason is that Ilango, Loff, and Oliveira [53] employ Equation (10.1) to design a learning algorithm for $\mathsf{SIZE}[n^a]$ that meets the requirements of Lemma 10.31 (see [53, Theorem 36, Item (2)]). Details follow.

Assume that Equation (10.1) holds. For $i \in \{0, 1, \ldots, t\}$ let $\mathcal{D}_i$ be the distribution supported over $\{0,1\}^{t(n+1)}$ that is obtained by sampling $x_1, b_1, \ldots, x_t, b_t$, where each x_j is a random n-bit string, and each b_j is set to $f(x_j)$ if $j > i$ and to a random bit otherwise. Let $p_i := \mathbf{Pr}_{w \sim \mathcal{D}_i, r}[B'(1^n, w, r) = 1]$. By Equation (10.1), we have

$$\left| \sum_{i=0}^{t-1} (p_i - p_{i+1}) \right| \geq \frac{1}{10}.$$

Therefore, there exists $j \in \{0, 1, \ldots, t-1\}$ such that

$$\left| \Pr_{w \sim \mathcal{D}_i, r} \left[B'(1^n, w, r) = 1 \right] - \Pr_{w \sim \mathcal{D}_i, r} \left[B'(1^n, w, r) = 1 \right] \right| \geq \frac{1}{10t}.$$

The only difference between the distributions $\mathcal{D}_j$ and $\mathcal{D}_{j+1}$ is that, for a random x_{j+1}, one generates the pair $(x_{j+1}, f(x_{j+1}))$ while the other generates the pair (x_{j+1}, b_{j+1}), where b_{j+1} is a random bit. All of the other coordinates of these two distributions are identically distributed

and can be generated by using the randomness of a (randomized) learner and the example oracle $\mathrm{EX}(f)$. Thus, if one knows $j+1$, it is possible to use B' or its negation to predict the value of f on a random input with advantage $\Omega(1/t)$ over a random guess [110].

This yields a randomized learning algorithm A that, with probability $\Omega(1/t)$ over its internal randomness and $\mathrm{EX}(f)$, outputs a circuit that is ε-close to f where $\varepsilon(n) = 1/2 - 1/(100t)$ [53]. The running time of A is polynomial under the assumption that B' runs in polynomial time. Note that A works infinitely often, by the assumption that B' makes Equation (10.1) hold (only) for infinitely many n.

The last step is to use the accuracy boosting technique by Boneh and Lipton [20] to improve the accuracy of A to $\varepsilon(n) = 1/\mathrm{poly}(n)$. This concludes the description of how a learning algorithm can be derived from Equation (10.1).

We now revisit Equation (10.1), and show that it holds. To this end, let B' be a probabilistic algorithm such that

$$B'(1^n, z, r) := B(1^n, (z_{n+1}, z_{2n+2}, \ldots, z_{t+tn}), (z_1, z_2, \ldots, z_n), (z_{n+2}, z_{n+3}, \ldots, z_{2n+1}), \\ \ldots, (z_{t+tn-n}, z_{t+tn-n+1}, \ldots, z_{t+tn-1}), r),$$

for all $n \in \mathbb{N}$, $z \in \{0,1\}^{t+tn}$, and $r \in \{0,1\}^{q(N)}$. That is, on input $(1^n, z)$ and using random bits r the algorithm B' applies a permutation σ on z to get a string $z' := \sigma(z) = z_{\sigma(1)} \cdots z_{\sigma(n)}$, and then runs B on $(1^n, z')$ using random bits r. For that matter B' runs in polynomial time, by the facts that σ is polynomial-time computable and B is a PPT algorithm.

We now have

$$\left| \underset{\{x_i\}_{i=1}^t, r}{\mathbf{Pr}} \left[B'(1^n, (x_1, f(x_1)), \ldots, (x_t, f(x_t)), r) = 1 \right] \right. \\ \left. - \underset{\{x_i\}_{i=1}^t, b, r}{\mathbf{Pr}} \left[B'(1^n, (x_1, b_1), \ldots, (x_t, b_t), r) = 1 \right] \right|$$

$$= \left| \underset{\{x_i\}_{i=1}^t, r}{\mathbf{Pr}} [B(1^n, (f(x_1), \ldots, f(x_t)), (x_1, \ldots, x_t), r) = 1] \right.$$
$$\left. - \underset{\{x_i\}_{i=1}^t, b, r}{\mathbf{Pr}} [B(1^n, b, (x_1, \ldots, x_t), r) = 1] \right| \geq \frac{1}{10}$$

by the definition of B', our assumption, and the fact that

$$\mathrm{KT}((f(x_1), \ldots, f(x_t)) \mid (x_1, \ldots, x_t)) \leq O\Big((n^a)^3\Big) \leq n^{4a} = n^{\gamma/c} = t^{1/c} = s < t/4,$$

by Lemma 10.11. □

10.1.10 Pseudorandom Functions

We will also require the notions of pseudorandom function families and distinguishers for function families.

Definition 10.24. Let $\{f_y\}_{y \in \{0,1\}^*}$ be a family of functions such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$. We say that the function family $\{f_y\}_{y \in \{0,1\}^*}$ is a *pseudorandom function family (PRF family)* if for all PPT oracle algorithms A there is a negligible function $\varepsilon : \mathbb{N} \to (0,1)$ such that for all sufficiently large $n \in \mathbb{N}$ it is the case that

$$\left| \underset{y \sim \{0,1\}^n, r}{\mathbf{Pr}} \Big[A^{f_y}(1^n, r) = 1 \Big] - \underset{g \sim \mathcal{F}_n, r}{\mathbf{Pr}} [A^g(1^n, r) = 1] \right| \leq \varepsilon(n)$$

where the size of r is equal to the running time of A.

Definition 10.25. Let $\{f_y\}_{y \in \{0,1\}^*}$ be a family of functions such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$. We say that a PPT oracle algorithm A is a *distinguisher for* $\{f_y\}_{y \in \{0,1\}^*}$ if for all negligible functions $\varepsilon : \mathbb{N} \to (0,1)$ and infinitely many $n \in \mathbb{N}$ it is the case that

$$\left| \underset{y \sim \{0,1\}^n, r}{\mathbf{Pr}} \Big[A^{f_y}(1^n, r) = 1 \Big] - \underset{g \sim \mathcal{F}_n, r}{\mathbf{Pr}} [A^g(1^n, r) = 1] \right| > \varepsilon(n)$$

where the size of r is equal to the running time of A.

Note how a distinguisher violates the guarantees of a PRF family. Below, we present a result by Goldreich, Goldwasser, and Micali [39], where they proved that the existence of PRGs implies the existence of PRFs.

Theorem 10.26 (PRGs imply PRFs; see Goldreich, Goldwasser, and Micali [39])**.** *If there exists a function $G = \{G_n\}_{n\in\mathbb{N}}$, whereby $G_n : \{0,1\}^{n/2} \to \{0,1\}^n$ for all $n \in \mathbb{N}$, such that for every PPT algorithm A there is a negligible function $\varepsilon : \mathbb{N} \to (0,1)$ such that G is a function that ε-fools A, then there exists a PRF family.*

Theorem 10.16 and Theorem 10.26 yield the following corollary.

Corollary 10.27 (OWFs imply PRFs)**.** *If there exists a OWF, then there exists a PRF family.*

10.2 OWFs From Average-case Hardness of McKTP

In this section, we prove the following result.

Theorem 10.28. *Assume that, for some $m : \mathbb{N} \to \mathbb{N}$, $\mathrm{McKT}^m\mathrm{P}$ of dimension n is $(1/p)$-HoA for some polynomial p. Then, there exists some weak OWF.*

By Theorem 10.28 and Theorem 10.18, we get the following corollary.

Corollary 10.29 (Theorem 1.15, restated)**.** *Assume that, for some $m : \mathbb{N} \to \mathbb{N}$, $\mathrm{McKT}^m\mathrm{P}$ of dimension n is $(1/p)$-HoA for some polynomial p. Then, there exists some OWF.*

We now prove Theorem 10.28.

Proof of Theorem 10.28. Fix some UTM U, and let $c > 0$ be as in Corollary 10.8. Let $n \in \mathbb{N}$ be sufficiently large and such that $\mathrm{McKT}^m\mathrm{P}$ of dimension n is $(1/p)$-HoA. Consider the function

$f : \{0,1\}^* \to \{0,1\}^*$ defined by the mapping rule

$$\left(s, t, y, \Pi'\right) \mapsto \left(s+t, U^{\Pi,y}\left(1, 1^t\right), \ldots, U^{\Pi,y}\left(n, 1^t\right), y\right),$$

where $m := m(n)$, $y \in \{0,1\}^m$, $\Pi' \in \{0,1\}^{n+c\log n}$ is a program, and $\Pi := \Pi'|_{[s]}$ is the s-bit prefix of Π'. Note that without loss of generality, $s+t \leq n+c\log n$, by Corollary 10.8. This also implies that $s \leq n + c\log n$ and $t \leq n + c\log n$. For that matter, f is a function from $\{0,1\}^M$ to $\{0,1\}^N$, where $M := 2\log\left(n + c\log n\right) + m + n + c\log n$ and $N := \log\left(n + c\log n\right) + n + m$, and is computable in polynomial time.

Observe also that f is only defined over infinitely many input lengths. However, by a padding trick, f can be transformed into another function f' that is defined over all input lengths, and such that f' is a weak one-way function, given that f is [70].

We now claim that if McKT^{m}P is $(1/p)$-HoA, then f is a $(1/q)$-weak OWF, where q is a polynomial such that $q(n) := 2\left(n + c\log n\right) n^c p(N')^3$ for all $n \in \mathbb{N}$, and $N' := n + m(n) + \log(n + c\log n)$ is the input size of McKT^{m}P. Towards a contradiction, assume that there exists a PPT algorithm A that inverts f with probability at least $1 - 1/q(M) \geq 1 - 1/q(n)$.

First, note that except for a fraction $1/\left(2p(N')\right)$ of sequences of random bits r for A, the deterministic machine A_r, given by $A_r(f(z)) := A(f(z), r)$ for all $z \in \{0,1\}^M$, fails to invert f with probability at most $2p(N')/q(n)$ over a uniformly random input z. This is so, as

$$\begin{aligned}
&\mathop{\mathbf{Pr}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A_r(f(z)) \text{ fails}] > \frac{2p(N')}{q(n)}\right] \\
&\quad\leq \mathop{\mathbf{Pr}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A_r(f(z)) \text{ fails}] \geq 2p\left(N'\right) \cdot \mathop{\mathbf{Pr}}_{z,r}[A_r(f(z)) \text{ fails}]\right] \\
&\quad= \mathop{\mathbf{Pr}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A(f(z), r) \text{ fails}] \geq 2p\left(N'\right) \cdot \mathop{\mathbf{E}}_{r}\left[\mathop{\mathbf{Pr}}_{z}[A(f(z), r) \text{ fails}]\right]\right] \\
&\quad\leq \frac{1}{2p(N')},
\end{aligned}$$

by Lemma 10.2. Henceforth, we will call such a sequence of random bits *good*; otherwise, we

will call a sequence of random bits *bad*. Therefore, we have

$$\mathop{\mathbf{Pr}}_{z,r}[A(f(z),r) \text{ fails} \mid r \text{ is good}] = \mathop{\mathbf{Pr}}_{z,r}[A_r(f(z)) \text{ fails} \mid r \text{ is good}] \leq \frac{2p(N')}{q(n)}.$$

We propose the following heuristic H for McKT^{m}P:

> On input strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, and a parameter $0 \leq \theta \leq n + c\log n$ in binary, and using random bits r, the algorithm H runs $A(j,x,y,r)$ for all $j \in [n + c\log n]$. For each $j \in [n + c\log n]$, $A(j,x,y,r)$ returns a tuple $\left(s_j, t_j, y, \Pi'_j\right)$. Then, $H(x,y,\theta,r)$ returns YES if there is a $j \in [n + c\log n]$ such that
>
> $$U^y\left(\Pi'_j\big|_{[s_j]}, i, 1^{t_j}\right) = x_i$$
>
> for all $1 \leq i \leq n$, and $\left|\Pi'_j\right|_{[s_j]}\Big| + t_j = s_j + t_j \leq \theta$. Otherwise, $H(x,y,\theta,r)$ returns NO.

Note that H runs in polynomial time. We will now analyze the average-case performance of H. Fix a good sequence of random bits r, as defined above, and recall that, in this case, $\mathbf{Pr}_z[A_r(f(z)) \text{ fails}] \leq 2p(N')/q(n)$. Let S_r be the set of inputs (x,y,θ) for which $H(x,y,\theta,r)$ fails, when given random bits r. Observe that, for any good r,

$$\mathop{\mathbf{Pr}}_{x,y,\theta}[H(x,y,\theta,r) \text{ fails}] = \frac{|S_r|}{2^{n+m} \cdot 2^{\log(n+c\log n)}} = \frac{|S_r|}{2^{n+m}\,(n + c\log n)}.$$

If $H(x,y,\theta,r)$ fails, then it means that A fails to invert (θ,x,y) when given the good sequence of random bits r.

Recall that $\mathbf{Pr}_z[A_r(f(z)) \text{ fails}] \leq 2p(N')/q(n)$. Recall also, from the definition of f, that

$$\mathop{\mathbf{Pr}}_{z}[f(z) = (\theta,x,y)] \geq \frac{1}{2^{2\log(n+c\log n)} \cdot 2^m \cdot 2^{n+c\log n}} = \frac{1}{(n + c\log n)^2 \cdot 2^m \cdot 2^{n+c\log n}}.$$

Thus, for any good sequence r, we have

$$\frac{2p(N')}{q(n)} \geq \mathop{\mathbf{Pr}}_{z}[A_r\left(f(z)\right) \text{ fails}]$$

$$
\begin{aligned}
&= \sum_{(\theta,x,y):A_r(\theta,x,y) \text{ fails}} \Pr_z[f(z) = (\theta,x,y)] \\
&\geq \sum_{(x,y,\theta)\in S_r} \frac{1}{(n+c\log n)^2 \cdot 2^m \cdot 2^{n+c\log n}} \\
&= \frac{|S_r|}{2^{n+m}\,(n+c\log n)} \cdot \frac{1}{(n+c\log n)\,2^{c\log n}} \\
&= \frac{\Pr_{x,y,\theta}[H(x,y,\theta,r) \text{ fails}]}{(n+c\log n)\,n^c}.
\end{aligned}
$$

Since this holds for any good sequence r, we have that

$$
\begin{aligned}
\Pr_{x,y,\theta,r}[H(x,y,\theta,r) \text{ fails} \mid r \text{ is good}] &\leq \frac{(n+c\log n)\,n^c 2p(N')}{q(n)} \\
&= \frac{(n+c\log n)\,n^c 2p(N')}{2\,(n+c\log n)\,n^c p(N')^3} \\
&= \frac{1}{p(N')^2} \\
&< \frac{1}{2p(N')},
\end{aligned}
$$

since $p(N') > 2$ for all sufficiently large $n \in \mathbb{N}$. Therefore, H fails with probability at most

$$
\Pr_{x,y,\theta,r}[H(x,y,\theta,r) \text{ fails} \mid r \text{ is good}] + \Pr_r[r \text{ is bad}] < \frac{1}{2p(N')} + \frac{1}{2p(N')} = \frac{1}{p(N')}.
$$

This yields a contradiction. □

10.3 Average-case Hardness of McKTP From OWFs

In this section, we prove the following result, which is a *weak* converse to Theorem 10.28.

Theorem 10.30 (Theorem 1.17, restated)**.** *Assume that there exists a OWF. Then, there exists a function $m : \mathbb{N} \to \mathbb{N}$ such that* $\mathrm{McKT}^m\mathrm{P}$ *of dimension n is $(1/h)$-HoA, for a function $h : \mathbb{N} \to \mathbb{R}_{>0}$ such that $h(N) := 2^{N/100}$ for all $N \in \mathbb{N}$.*

10.3.1 Applications of Heuristics

We will require the following result, which asserts that heuristics with good average-case performance guarantees can be used to design learning algorithms.

Lemma 10.31. *If for all* $0 < \gamma < 1$ *it is the case that* McKT^{m}P *of dimension* n *and* $m := n^{1+\gamma}$ *is* $(1 - 1/h)$*-EoA, for a function* $h : \mathbb{N} \to \mathbb{R}_{>0}$ *such that* $h(N) := 2^{N/100}$ *for all* $N \in \mathbb{N}$*, then for every* $a \geq 1$ *the class* $\mathsf{SIZE}[n^a]$ *can be infinitely often learned in polynomial time with accuracy error* $\varepsilon = 1/n$ *and confidence error* $\delta = 1/n$*.*

Proof. We will apply Lemma 10.23, by proving that if for all $0 < \gamma < 1$ it is the case that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is $(1 - 1/h)$-EoA, then for all $c > 1$ and all $0 < \gamma < 1$ it is the case that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is EoA with advantage $1/10$ for a size parameter $s := n^{1/c}$ and infinitely many $n \in \mathbb{N}$. The desired result will then follow from Lemma 10.23.

Let $0 < \gamma < 1$ be arbitrary. Let H be the heuristic that witnesses the fact that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is $(1 - 1/h)$-EoA (see Definition 10.20), and assume that H runs in time $q(N)$ for some polynomial q.

Let $n \in \mathbb{N}$ be sufficiently large and such that H satisfies its average-case performance guarantees on inputs of dimension n. In what follows, let $N := n + m(n) + \log(n + \sigma \log n)$ be the size of the McKT^{m}P on instances of dimension n, where σ is from Corollary 10.8; see Definition 10.12.

Let $c > 1$ be arbitrary, and $s := n^{1/c}$. Let H^* be a probabilistic algorithm such that, for all strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^{m(n)}$, and all random strings $r \in \{0,1\}^{q(N)}$, satisfies $H^*(1^n, x, y, r) := H(x, y, s, r)$.

By the definition of H^*, we have that the first term of the LHS of the inequality of

Definition 10.21 is

$$\mathop{\mathbf{Pr}}_{y,r}[H^*(1^n, x, y, r) = 1] = \mathop{\mathbf{Pr}}_{y,r}[H(x, y, s, r) = 1]\,,$$

where $y \in \{0,1\}^{m(n)}$ is uniformly random, and $x = x(y) \in \{0,1\}^n$ is such that $\mathrm{KT}(x \mid y) \leq s$, as guaranteed to exist by Lemma 10.9. We claim that this probability is sufficiently large.

Claim 10.32. It is the case that

$$\mathop{\mathbf{Pr}}_{y,r}[H(x, y, s, r) = 1] \geq \frac{1}{5}.$$

Proof. Towards a contradiction, assume that

$$\mathop{\mathbf{Pr}}_{y,r}[H(x, y, s, r) = 1] < \frac{1}{5}.$$

Then, the number of inputs and random bits of H that make H output "0" is greater than

$$K := 2^{m(n)} \cdot 2^{q(N)} \cdot \left(1 - \frac{1}{5}\right) = 2^{m(n)+q(N)} \cdot \frac{4}{5}.$$

However, it is the case that $K < 2^{N+q(N)}/h(N)$, by our assumption on the average-case performance of H. Thus

$$2^{m(n)+q(N)} \cdot \frac{4}{5} < \frac{1}{h(N)} \cdot 2^{N+q(N)} = \frac{1}{2^{N/100}} \cdot 2^{n+m(n)+\log(n+\sigma \log n)+q(N)}$$

or

$$\frac{4}{5} < \frac{1}{2^{N/100}} \cdot 2^{n+\log(n+\sigma \log n)} \leq \frac{1}{2^{n^{1+\gamma}/100}} \cdot 2^{2n} < \frac{4}{500};$$

this yields a contradiction. (Claim 10.32) □

We now turn to the second term of the LHS of the inequality of Definition 10.21. To this end, we have

$$\mathop{\mathbf{Pr}}_{z,y,r}[H^*(1^n, z, y, r) = 1] = \mathop{\mathbf{Pr}}_{z,y,r}[H(z, y, s, r) = 1]$$

$$\begin{aligned}
&\leq \mathop{\mathbf{Pr}}_{z,y,r}[H(z,y,s,r)=1 \mid \mathrm{KT}(z \mid y) > s] + \mathop{\mathbf{Pr}}_{z,y}[\mathrm{KT}(z \mid y) \leq s] \\
&= \frac{\mathbf{Pr}_{z,y,r}[H(z,y,s,r)=1 \text{ and } \mathrm{KT}(z \mid y) > s]}{\mathbf{Pr}_{z,y}[\mathrm{KT}(z \mid y) > s]} + \mathop{\mathbf{Pr}}_{z,y}[\mathrm{KT}(z \mid y) \leq s] \\
&\leq \frac{1/h(N)}{1/2} + \frac{1}{20},
\end{aligned}$$

by Lemma 10.10, or

$$\begin{aligned}
&= \frac{2}{h(N)} + \frac{1}{20} \\
&= \frac{2}{2^{N/100}} + \frac{1}{20} \\
&\leq \frac{2}{2^{n/100}} + \frac{1}{20} \\
&\leq \frac{1}{20} + \frac{1}{20} \\
&= \frac{1}{10},
\end{aligned}$$

Therefore, by Claim 10.32 and the discussion above,

$$\mathop{\mathbf{Pr}}_{y,r}[H^*(1^n,x,y,r)=1] - \mathop{\mathbf{Pr}}_{z,y,r}[H^*(1^n,z,y,r)] \geq \frac{1}{5} - \frac{1}{10} = \frac{1}{10},$$

as desired. □

Remark 10.33. It should be noted that Lemma 10.31 also holds for the case where McKTP is *zero-error* easy on average.

10.3.2 Applications of Infinitely Often Learning Algorithms

An important observation is that a learning algorithm for polynomial-size circuits may be used to create distinguishers for polynomial-time computable function families.

Lemma 10.34 (See also Oliveira and Santhanam [87, Theorem 8])**.** *Assume that for every* $a \geq 1$ *the class* $\mathsf{SIZE}[n^a]$ *can be infinitely often learned in polynomial time with accuracy error*

$\varepsilon = 1/n$ and confidence error $\delta = 1/n$. Then, for all function families $\{f_y\}_{y\in\{0,1\}^*}$ such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$, there is a distinguisher for $\{f_y\}_{y\in\{0,1\}^*}$.

Proof. Let $\{f_y\}_{y\in\{0,1\}^*}$ be a function family such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$. In particular, for all $y \in \{0,1\}^*$ it is the case that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ is computable in time $|y|^{k/2}$ for some $k > 0$. By Lemma 10.6, for all $y \in \{0,1\}^*$ it is the case that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ is computable by some circuit of size

$$O\Big(|y|^{k/2} \log |y|\Big) \leq |y|^k .$$

Let A be the PPT learning algorithm for $\mathsf{SIZE}\big[n^k\big]$ that works for infinitely many $n \in \mathbb{N}$, and that runs in time $q(n)$ for some polynomial q, as guaranteed to exist by our assumption. In particular, let $n \in \mathbb{N}$ be sufficiently large and such that A satisfies its learning guarantees on inputs of size n. Let $t := n^{100}$ and define D to be a probabilistic oracle algorithm as follows.

> On input 1^n, random bits $r := (r', r'') \in \{0,1\}^{O(q(n))} \times \{0,1\}^{t\cdot n}$, and given oracle access to some function $g : \{0,1\}^n \to \{0,1\}$, the algorithm D runs A on 1^n using random bits $r' \in \{0,1\}^{O(q(n))}$ to get a hypothesis h for g, whereby simulating calls to $\mathrm{EX}(g)$ by using the oracle for g. Then, D samples t strings $x_1, \ldots, x_t$ from $\{0,1\}^n$ using random bits $r'' \in \{0,1\}^{t\cdot n}$ and uses the oracle for g to compute
>
> $$\alpha := \frac{|\{i \in [t] \mid h(x_i) = g(x_i)\}|}{t}.$$
>
> Finally, if $\alpha \geq 2/3$, then D outputs 1; else, D outputs 0.

Note that D runs in time polynomial in n. We will now prove that D is a distinguisher for $\{f_y\}_{y\in\{0,1\}^*}$. To this end, we will show that D satisfies Definition 10.25.

The first term of the LHS of the inequality of Definition 10.25 is

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{y\sim\{0,1\}^n,r}\left[D^{f_y}(1^n,r)=1\right] &= \mathop{\mathbf{Pr}}_{y,r}\left[\alpha \geq \frac{2}{3}\right] \\
&= \mathop{\mathbf{Pr}}_{y,r}\left[\frac{|\{i\in[t] \mid h(x_i)=f_y(x_i)\}|}{t} \geq \frac{2}{3}\right] \\
&\geq \mathop{\mathbf{Pr}}_{y,r}\left[\frac{|\{i\in[t] \mid h(x_i)=f_y(x_i)\}|}{t} \geq \frac{3}{4}\left(1-\frac{1}{n}\right)\right] \\
&= \mathop{\mathbf{Pr}}_{y,r}\left[\frac{|\{i\in[t] \mid h(x_i)=f_y(x_i)\}|}{t} \geq \frac{3}{4}\left(1-\varepsilon\right)\right] \\
&\geq \mathop{\mathbf{Pr}}_{y,r}\left[\frac{|\{i\in[t] \mid h(x_i)=f_y(x_i)\}|}{t} \geq \frac{3}{4}\left(1-\varepsilon\right) \mid h \text{ is } \varepsilon\text{-close to } f_y\right] \\
&\qquad \cdot \mathop{\mathbf{Pr}}_{y,r}[h \text{ is } \varepsilon\text{-close to } f_y]\,.
\end{aligned}$$

For all $1 \leq i \leq t$, let X_i be a Boolean variable such that $X_i := 1$ if and only if $h(x_i) = f_y(x_i)$. Then,

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{y,r}\left[D^{f_y}(1^n,r)=1\right] &\geq \mathop{\mathbf{Pr}}_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t} \geq \frac{3}{4}\left(1-\varepsilon\right) \mid h \text{ is } \varepsilon\text{-close to } f_y\right] \cdot \mathop{\mathbf{Pr}}_{y,r}[h \text{ is } \varepsilon\text{-close to } f_y] \\
&\geq \mathop{\mathbf{Pr}}_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t} \geq \frac{3}{4}\mathop{\mathbf{E}}_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t}\right]\right]\left(1-\delta\right),
\end{aligned}$$

as $\mathrm{CC}(f_y) \leq |y|^k = n^k$, or

$$\geq \frac{1}{4}\mathop{\mathbf{E}}_{y,r}\left[\frac{\sum_{i=1}^t X_i}{t}\right]\left(1-\delta\right),$$

by Lemma 10.1, or

$$\begin{aligned}
&\geq \frac{1}{4}\left(1-\varepsilon\right)\left(1-\delta\right) \\
&= \frac{1}{4}\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n}\right) \\
&\geq \frac{1}{4}-\frac{1}{8} \\
&= \frac{1}{8}.
\end{aligned}$$

We now turn to the second term of the LHS of the inequality of Definition 10.25. To this end, we have

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{g\sim\mathcal{F}_n,r}[D^g(1^n,r)=1] &= \mathop{\mathbf{Pr}}_{g,r}\left[\alpha \geq \frac{2}{3}\right] \\
&= \mathop{\mathbf{Pr}}_{g,r}\left[\frac{|\{i\in[t] \mid h(x_i)=g(x_i)\}|}{t} \geq \frac{2}{3}\right] \\
&\leq \mathop{\mathbf{Pr}}_{g,r}\left[\frac{|\{i\in[t] \mid h(x_i)=g(x_i)\}|}{t} \geq \frac{2}{3} \mid \{x_i\}_{i=1}^t \text{ are distinct}\right] \\
&\quad + \mathbf{Pr}\left[\{x_i\}_{i=1}^t \text{ are not distinct}\right] \\
&= \mathop{\mathbf{Pr}}_{b,r}\left[|\{i\in[t] \mid h(x_i)=b_i\}| \geq \frac{2t}{3} \mid \{x_i\}_{i=1}^t \text{ are distinct}\right] \\
&\quad + \mathbf{Pr}\left[\{x_i\}_{i=1}^t \text{ are not distinct}\right],
\end{aligned}$$

where $b_1,\ldots,b_t \in \{0,1\}$ are independent and uniformly random, and $b := (b_1,\ldots,b_t)$.

Similarly as above, for all $1 \leq i \leq t$, let X_i be a Boolean variable such that $X_i := 1$ if and only if $h(x_i) = b_i$. Let Y be the event that all of the x_i are distinct. Then,

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{g\sim\mathcal{F}_n,r}[D^g(1^n,r)=1] &\leq \mathop{\mathbf{Pr}}_{b,r}\left[\sum_{i=1}^t X_i \geq \frac{2t}{3} \mid Y\right] + \mathbf{Pr}\left[\{x_i\}_{i=1}^t \text{ are not distinct}\right] \\
&= \mathop{\mathbf{Pr}}_{b,r}\left[\sum_{i=1}^t X_i \geq \frac{4}{3}\cdot\frac{t}{2} \mid Y\right] + \mathbf{Pr}[\exists i,j\in[t] : i\neq j \text{ and } x_i = x_j] \\
&\leq \mathop{\mathbf{Pr}}_{b,r}\left[\sum_{i=1}^t X_i \geq \frac{4}{3}\cdot\frac{t}{2} \mid Y\right] + \binom{t}{2}2^{-n},
\end{aligned}$$

by a union bound, or

$$\begin{aligned}
&= \mathop{\mathbf{Pr}}_{b,r}\left[\sum_{i=1}^t X_i \geq \left(1+\frac{1}{3}\right)\mathop{\mathbf{E}}_{b,r}\left[\sum_{i=1}^t X_i \mid Y\right] \mid Y\right] + \binom{n^{100}}{2}2^{-n} \\
&\leq \left(e^{-\frac{1/9}{3}}\right)^{\mathbf{E}_{b,r}\left[\sum_{i=1}^t X_i \mid Y\right]} + \frac{n^{200}}{2^n},
\end{aligned}$$

by Lemma 10.3, or

$$\leq \left(e^{-1/27}\right)^{t/2} + \frac{1}{32}$$

$$
\begin{aligned}
&= e^{-t/54} + \frac{1}{32} \\
&= e^{-n^{100}/54} + \frac{1}{32} \\
&\leq \frac{1}{32} + \frac{1}{32} \\
&= \frac{1}{16}.
\end{aligned}
$$

Therefore,

$$\left| \mathop{\mathbf{Pr}}_{y \sim \{0,1\}^n, r} \left[D^{f_y}(1^n, r) = 1 \right] - \mathop{\mathbf{Pr}}_{g \sim \mathcal{F}_n, r} [D^g(1^n, r) = 1] \right| \geq \frac{1}{8} - \frac{1}{16} = \frac{1}{16} = \Omega(1)$$

and as every negligible function $\mu : \mathbb{N} \to [0,1]$ is such that $\mu(n) = o(1) < \Omega(1)$, the desired result follows. □

10.3.3 Proof of Theorem 10.30

We will prove the contrapositive. To this end, assume that for all functions $m : \mathbb{N} \to \mathbb{N}$ it is the case that McKT^{m}P of dimension n is $(1 - 1/h)$-EoA, for a function $h : \mathbb{N} \to \mathbb{R}_{>0}$ such that $h(N) := 2^{N/100}$ for all $N \in \mathbb{N}$. This implies that for all $0 < \gamma < 1$ it is the case that McKT^{m}P of dimension n and $m := n^{1+\gamma}$ is $(1 - 1/h)$-EoA, for a function $h : \mathbb{N} \to \mathbb{R}_{>0}$ such that $h(N) := 2^{N/100}$ for all $N \in \mathbb{N}$.

By Lemma 10.31, for all $a \geq 1$, we get a learning algorithm for $\mathsf{SIZE}[n^a]$ that works for infinitely many $n \in \mathbb{N}$. By Lemma 10.34, for every function family $\{f_y\}_{y \in \{0,1\}^*}$ such that $f_y : \{0,1\}^{|y|} \to \{0,1\}$ for all $y \in \{0,1\}^*$, and there is a polynomial-time algorithm that computes $f_y(x)$ given y and $x \in \{0,1\}^{|y|}$, there is a distinguisher for $\{f_y\}_{y \in \{0,1\}^*}$. By Corollary 10.27, there are no OWFs.

10.4 McKTP is NP-complete Under Randomized Reductions

In this section, we prove Theorem 1.16 by adapting Ilango's work [52]. Ilango [52] showed that the Minimum Oracle Circuit Size Problem (MOCSP) is NP-hard by capitalizing on the facts that a finite oracle O can encode useful information about an *optimal* set cover, and that approximating Set Cover is NP-hard [34]. Then, our result follows from the fact that a finite oracle can be encoded by a finite string y (for example, y can be the truth table of O), that can serve the role of the *auxiliary* string in the definition of McKTP. Details follow by adapting Ilango [52].

Suppose that we are given sets $S_1, \ldots, S_t \subseteq [n]$ such that $\bigcup_{i=1}^{t} S_i = [n]$. Then, for a uniformly random truth table T of length $m \geq (nt)^5$, we let each set from $\{S_i\}_{i=1}^{t}$ induce truth tables $T_{S_1}, \ldots, T_{S_t}$ such that each truth table T_{S_i} has the size of T, and "sees" roughly the same part of T as S_i "sees" of $[n]$. We define our auxiliary string y to be the concatenation of these truth tables. Finally, we ask how hard it is for a program to compute T efficiently given the auxiliary string y.

The intuition is that if a program looks at the collection of induced truth tables corresponding to a set cover, then it can "see" all of T and thus easily compute T. If not, then a large part of T is still random to the program and thus hard to compute. That is, we show that with high probability the quantity $\mathrm{KT}(T \mid y)$ can be used to estimate the size of an optimal set cover up to a constant factor.

We first give some background information.

10.4.1 Approximation Algorithms

In the following, we will adopt the following notion of an approximation algorithm.

Definition 10.35. Let Π be an optimization problem. For all instances $I \in \{0,1\}^*$ of Π, let the optimal solution of I be denoted by $\mathrm{OPT}(I) \in \mathbb{R}$. Let $\alpha > 0$. We say that a probabilistic algorithm A *approximates* Π *by a factor of* α if, for all instances I of Π, it is the case that

$$\mathrm{OPT}(I) < A(I) \leq \alpha \cdot \mathrm{OPT}(I)$$

with probability at least $1 - o(1)$ over the internal randomness of A.

10.4.2 Set Cover

We first fix some notation about Set Cover.

Definition 10.36. The *Set Cover* problem is defined as follows.

- Input: A tuple $(n, S_1, \ldots, S_t)$ encoded in binary, where $n \in \mathbb{N}$ and $S_1, \ldots, S_t \subseteq [n]$ are sets such that $[n] \subseteq \bigcup_{i=1}^{t} S_i$.
- Output: The minimum value of $|I|$, over the possible choices of $I \subseteq [t]$, such that

$$[n] \subseteq \bigcup_{i \in I} S_i.$$

Dinur and Steurer [34] show that it is NP-hard to approximate Set Cover.

Theorem 10.37 ([34])**.** *It is* NP*-hard to approximate Set Cover by a factor of at most* $(1 - o(1)) \ln n$.

10.4.3 Proof of Theorem 1.16

For a string b of length m and a set $R \subseteq [m]$, let $b_{\langle R \rangle}$ be the string of length m where

$$b_{\langle R \rangle}(j) := \begin{cases} b(j) & \text{if } j \in R, \\ 0 & \text{otherwise} \end{cases}$$

for all $1 \leq j \leq m$. Equivalently,

$$b_{\langle R \rangle}(j) := b(j) \wedge \mathbb{1}_{j \in R}$$

for all $j \in [m]$.

Next, we define a uniformly random partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ into n parts to be such that each element $i \in [m]$ is put into P_j where $j \in [n]$ is chosen uniformly at random. It will be also useful to think of $\mathcal{P}$ as a uniformly random function $P : [m] \to [n]$.

For a partition $\mathcal{P} = (P_1, \ldots, P_n)$ of $[m]$ and any set $S \subseteq [n]$, we define the *$\mathcal{P}$-lift of S*, denoted $S^{\mathcal{P}}$, to be the set

$$S^{\mathcal{P}} := \bigcup_{i \in S} P_i.$$

Following Ilango [52], we show that McKTP can be used to approximate Set Cover.

Lemma 10.38 (Following Ilango [52])**.** *Let $S_1, \ldots, S_t \subseteq [n]$ be sets that cover $[n]$. Let b be a string of length $m \geq (nt)^5$ and let $\mathcal{P} = (P_1, \ldots, P_n)$ be a uniformly random partition of $[m]$ into n parts. Define the oracle $O : \{0,1\}^{\log t} \times \{0,1\}^{\log m} \to \{0,1\}$ to be such that*

$$O(i, z) := \begin{cases} b_{\langle S_i^{\mathcal{P}} \rangle}(z) & \text{if } i \in [t], \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in [t]$ and $z \in [m]$. Let y be the truth table of O, and note that $|y| = mt$. Let ℓ be the size of the optimal cover of $[n]$ by $S_1, \ldots, S_t$. Then, we have that

1. $\mathrm{KT}(b \mid y) \leq 200\ell \left(\log t + \log m\right)$ *and*
2. $\mathrm{KT}(b \mid y) > \ell \left(\log t + \log m\right)/2$ *with high probability over the choice of b.*

Proof. We prove each item of Lemma 10.38 separately.

Claim 10.39. It is the case that $\mathrm{KT}(b \mid y) \leq 200\ell \left(\log t + \log m\right)$.

Proof. Assume that an optimal set cover of size ℓ is realized by the sets $S_{i_1}, \ldots, S_{i_\ell}$. Fix some UTM U that has oracle access to y. Let $\Pi \in \{0,1\}^*$ be a program that contains in its description encodings of $i_1, \ldots, i_\ell \in \{0,1\}^t$ and operates as follows:

> On input $x \in \{0,1\}^{\log m}$, compute and output $y_{(i_1,x)} \vee \cdots \vee y_{(i_\ell,x)}$.

Note that $|\Pi| \leq (\ell+2)\log t + O(1) \leq 100\ell \log t$. In what follows, let $T \in \mathbb{N}$ be a sufficiently large run-time bound such that

$$\begin{aligned} U^{\Pi,y}\left(x, 1^T\right) &:= y_{(i_1,x)} \vee \cdots \vee y_{(i_\ell,x)} \\ &= O(i_1, x) \vee \cdots \vee O(i_\ell, x) \\ &= \bigvee_{i \in [\ell]} \bigvee_{j \in S_i} b_{\langle P_j \rangle}(x) \\ &= \bigvee_{j \in [n]} b_{P_j}(x) \\ &= b(x)\,, \end{aligned}$$

for all $x \in \{0,1\}^{\log m}$. Note that $T \leq 100\ell\,(\log t + \log m)$. Therefore, it is the case that $\mathrm{KT}(b \mid y) \leq 200\ell\,(\log t + \log m)$. (Claim 10.39) □

We now turn to the lower bound. We do this by a union bound argument. Fix some oracle program $M^y(\cdot) := U^{\Pi,y}\left(\cdot, 1^T\right)$ of program Π that uses oracle y and runs in time T such that $|\Pi| + T \leq \ell\,(\log t + \log m)\,/2$. Then, as each oracle query requires time $\log t + \log m$, we can deduce that M makes at most $\ell/2 \leq n/2 \leq n$ oracle queries to y.

We will show that

$$\mathop{\mathbf{Pr}}_{b,\mathcal{P}}[M^y \text{ computes } b \text{ in time } T, \text{ and } |\Pi| + T \leq \ell\,(\log t + \log m)\,/2]$$

is exponentially small. We do this by finding a long sequence of inputs $x_1, \ldots, x_d$ on which M has not too large a chance of computing b.

We construct this list recursively, as follows. Let $x_1 := 0^{\log m}$, and let

$$Q_1 := \left\{ x \in \{0,1\}^{\log m} \mid M^y(x_1) \text{ makes a query of the form } (i,x) \text{ to } y, \text{ for some } i \in [t] \right\}.$$

Now, for $j \geq 1$, if $\{0,1\}^{\log m} \setminus Q_j$ is non-empty, then let x_{j+1} be an element of $\{0,1\}^{\log m} \setminus Q_j$, and let

$$Q_{j+1} := Q_j \cup \left\{ x \in \{0,1\}^{\log m} \mid M^y(x_{j+1}) \text{ makes a query of the form } (i,x), \text{ for some } i \in [t] \right\}.$$

If $\{0,1\}^{\log m} = Q_j$, then terminate the sequence. Since M makes at most n queries to y, we know that $|Q_j| \leq jn$. Thus, since

$$|Q_d| = \left| \{0,1\}^{\log m} \right| = m$$

the length of this sequence is at least m/n. That is, $d \geq m/n$.

It remains to bound the probability

$$\mathbf{Pr}[\text{for all } j \in [d],\ M^y(x_j) = b(x_j)] = \prod_{j=1}^{d} \mathbf{Pr}\left[M^y(x_j) = b(x_j) \mid \bigwedge_{k \in [j-1]} M^y(x_k) = b(x_k) \right].$$

Fix some $j \in [d]$. We will bound

$$\mathbf{Pr}\left[M^y(x_j) = b(x_j) \mid \bigwedge_{k \in [j-1]} M^y(x_k) = b(x_k) \right].$$

Let $E := \bigwedge_{k \in [j-1]} M^y(x_k) = b(x_k)$ be the event that we are conditioning on.

Claim 10.40. It is the case that

$$\mathbf{Pr}[M^y(x_j) = b(x_j) \mid E] \leq 1 - \frac{1}{2n}.$$

Proof. By construction of the sequence $x_1, \ldots, x_d$, we know that on all the inputs $x_1, \ldots, x_{j-1}$, the program M^y does not make an oracle call of the form (i, x_j) for any i. Thus, the only time the value of O depends on $b(x_j)$ and $P(x_j)$ is on inputs of the form (i, x_j) for some i, and since

$b(x_j)$ and $P(x_j)$ are chosen independently at random, we know that $b(x_j)$ and $P(x_j)$ are still uniform random variables conditioned on E. That is,

$$\mathbf{Pr}[b(x_j) = 1 \mid E] = \frac{1}{2}$$

and

$$\mathbf{Pr}[P(x_j) = r \mid E] = \frac{1}{n}$$

for all $r \in [n]$.

Now, define O' as

$$O'(i,x) := \begin{cases} 0 & \text{if } x = x_j, \\ O(i,x) & \text{otherwise} \end{cases}$$

and let y' be the truth table of O'. Let also $i_1, \ldots, i_v$ with $v \leq \ell/2$ be such that, using the modified oracle O', they are the only oracle queries $M^{y'}(x_j)$ makes that have x_j as the 2nd component of the query, so the queries are $(i_1, x_j), \ldots, (i_v, x_j)$. Since $v < \ell$ there exists an element r^* that is not in $S_{i_1} \cup \cdots \cup S_{i_v}$.

Moreover, observe that if $P(x_j) = r^*$, then $M^y(x_j)$ will actually make the same oracle queries (and get the same zero responses) as the modified oracle program $M^{y'}$. In this case, since $P(x_j) = r^*$ is not in $S_{i_1} \cup \cdots \cup S_{i_v}$, it follows that

$$O(i_1, x_j) = \cdots = O(i_v, x_j) = 0$$

regardless of the value of $b(x_j)$. Thus, the output of M^y on input x does not depend at all on the value of $b(x)$ if $P(x_j) = r^*$. Hence, the probability it correctly guesses $M^y(x) = b(x)$ is at most half when $P(x_j) = r^*$.

Since $P(x_j)$ is chosen uniformly at random, we have that $P(x_j) = r^*$ with probability $1/n$. Therefore,

$$\mathbf{Pr}[M^y(x_j) = b(x_j) \mid E] \leq 1 - \frac{1}{2n}.$$ (Claim 10.40) □

Using Claim 10.40, we have

$$\begin{aligned}\prod_{j=1}^{d} \mathbf{Pr}\left[M^y(x_j) = b(x_j) \mid \bigwedge_{k\in[j-1]} M^y(x_k) = b(x_k)\right] &\leq \left(1 - \frac{1}{2n}\right)^d \\ &\leq e^{-d/(2n)} \\ &\leq e^{-m/\left(2n^2\right)} \\ &\leq e^{-n^3t^5/2}.\end{aligned}$$

On the other hand, the number of oracle programs of size at most $\ell\left(\log t + \log m\right)/2 \leq O(nt \log n)$ is at most $2^{O\left(n^2t\right)}$. Thus, by a union bound, the probability that there exists an oracle program Π that computes any bit of b in time T, whereby $|\Pi| + T \leq \ell\left(\log t + \log m\right)/2$, is $o(1)$ as desired. □

Lemma 10.38 implies the following corollary.

Corollary 10.41. *There is a polynomial-time computable function $M : \mathbb{N} \to \mathbb{N}$ such that the following holds. Given a Set Cover instance $I := (n, S_1, \ldots, S_t)$, a random b of length $N \geq (nt)^5$ and a random partition P of $[N]$ into n parts, if one constructs a string y as in Lemma 10.38, whereby $|y| \leq M(N)$, then $\mathrm{KT}(b \mid y)$ approximates Set Cover by a factor of 400 according to Definition 10.35. That is, if ℓ is the size of the optimal set cover of I and $c := \log N + \log t$, then it is the case that with probability 1*

$$\frac{2}{c} \cdot \mathrm{KT}(b \mid y) \leq 400\ell,$$

and with probability $1 - o(1)$

$$\frac{2}{c} \cdot \mathrm{KT}(b \mid y) > \ell.$$

Proof. Let $y \in \{0,1\}^*$, $n \in \mathbb{N}$, and $t \in \mathbb{N}$ be as in Lemma 10.38. Let $\gamma := 1/2$. Then, $\mathrm{McKT}^M\mathrm{P}$ of dimension $N := |b| \geq (nt)^5$ and $M := N^{1+\gamma} = N^{1+1/2} = N \cdot N^{1/2} \geq Nt = |y|$ is such that

Lemma 10.38 immediately implies that

$$\ell < \frac{2}{c} \cdot \mathrm{KT}(b \mid y) \leq 400\ell,$$

where the first inequality holds with probability $1 - o(1)$ and the second one holds with probability 1. □

Theorem 10.37 and Corollary 10.41 yield the following corollary.

Corollary 10.42. *There exists a polynomial-time computable function $m : \mathbb{N} \to \mathbb{N}$ such that* $\mathrm{McKT}^m\mathrm{P}$ *is* NP*-hard under polynomial-time randomized reductions.*

Finally, by combining Lemma 10.13 and Corollary 10.42 we get a proof of Theorem 1.16.

Corollary 10.43 (Theorem 1.16, restated)**.** *There exists a polynomial-time computable function $m : \mathbb{N} \to \mathbb{N}$ such that* $\mathrm{McKT}^m\mathrm{P}$ *is* NP*-complete under polynomial-time randomized reductions.*

Notes

The results of this chapter are products of joint work with Eric Allender, Mahdi Cheraghchi, Harsha Tirumala, and Ilya Volkovich [7].

Chapter 11

Discussion

Our work establishes many new lower bounds for MCSP and MKTP against concrete models of computation, all of which almost match the state-of-the-art lower bounds against these models and are the first of their kind for MCSP and MKTP. We also showed an interesting connection between the average-case complexity of a variant of MKTP and the existence of OWFs.

11.1 Open Problems

We believe that this line of work leaves some interesting open problems, for future work, as follows.

1. As mentioned above, our work establishes new lower bounds for MCSP against many models of computation. These lower bounds *almost* match the state-of-the-art lower bounds against these models of computation. Can our MCSP lower bounds be improved to exactly match the state-of-the-art lower bounds?

2. Are there alternative proofs to prove our MCSP lower bounds, that do *not* rely on local PRGs? For example, can we apply Nečiporuk's method to MCSP?

3. What are other restricted models of computation against which we can show MCSP lower bounds using local PRGs?

   The recent "random walk" PRG by Chattopadhyay, Hatami, Hosseini, and Lovett [24] is also local and can be used to get MCSP lower bounds. However, as a general PRG that can be used to fool a variety of restricted models it has sub-optimal usefulness (which is determined by its seed length) compared to the best-known lower bounds for most of those models.

4. Is it possible to adapt existing techniques to show an explicit lower bound against $\mathsf{FORMULA}[n^{2.01}] \circ \mathsf{XOR}$, or achieving this is just as hard as breaking the cubic barrier for formula lower bounds?

5. In this thesis, we established a weak equivalence between the existence of OWFs and the average-case hardness of a variant of MKTP that is NP-complete. Is there such an equivalence between meta-complexity problems and other cryptography primitives, especially primitives whose existence is not known to be implied by OWFs, such as public key cryptosystems?

Bibliography

[1] Amir Abboud and Karl Bringmann. Tighter connections between formula-sat and shaving logs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 8:1–8:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 99–108. ACM, 1996.

[3] Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. *Advances in Computing Research*, 5:199–222, 1989.

[4] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 701–710. ACM, 2006. See also [5].

[5] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. Erratum for: On basing one-way functions on NP-hardness. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 795–796. ACM, 2010.

[6] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[7] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and a conditional variant of MKTP. Manuscript, 2021.

[8] Eric Allender, Mahdi Cheraghchi, Dimitrios Myrisiotis, Harsha Tirumala, and Ilya Volkovich. One-way functions and Partial MCSP. *Electron. Colloquium Comput. Complex.*, 28:9, 2021.

[9] Eric Allender, Joshua A. Grochow, Dieter van Melkebeek, Cristopher Moore, and Andrew Morgan. Minimum circuit size, graph isomorphism, and related problems. *SIAM J. Comput.*, 47(4):1339–1372, 2018.

[10] Eric Allender and Shuichi Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[11] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[12] Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. *J. Comput. Syst. Sci.*, 77(1):14–40, 2011.

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

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

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

[16] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.

[17] Paul Beame, Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. Nondeterminism and an abstract formulation of Nečiporuk's lower bound method. *ACM Trans. Comput. Theory*, 9(1):5:1–5:34, 2016.

[18] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006.

[19] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[20] Dan Boneh and Richard J. Lipton. Amplification of weak learning under the uniform distribution. In Lenny Pitt, editor, *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993*, pages 347–351. ACM, 1993.

[21] Chris Brzuska and Geoffroy Couteau. Towards fine-grained one-way functions from strong average-case hardness. *IACR Cryptol. ePrint Arch.*, 2020:1326, 2020.

[22] Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory Comput. Syst.*, 40(4):379–395, 2007.

[23] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 10:1–10:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[24] Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 1:1–1:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[25] Lijie Chen, Shuichi Hirahara, Igor Carboni Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond natural proofs: Hardness magnification and locality. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 70:1–70:48, 2020.

[26] Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1240–1255, 2019.

[27] 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.

[28] Mahdi Cheraghchi, Shuichi Hirahara, Dimitrios Myrisiotis, and Yuichi Yoshida. One-tape Turing machine and branching program lower bounds for MCSP. In Markus Bläser and Benjamin Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference)*, volume 187 of *LIPIcs*, pages 23:1–23:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[29] Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. *ACM Trans. Comput. Theory*, 12(3):21:1–21:27, 2020.

[30] Stephen A. Cook and Robert A. Reckhow. Time-bounded random access machines. In Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg, editors, *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 73–80. ACM, 1972.

[31] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC^0. In Jirí Sgall, Ales Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001, 26th International Symposium, MFCS 2001 Marianske Lazne, Czech Republic, August 27-31, 2001, Proceedings*, volume 2136 of *Lecture Notes in Computer Science*, pages 272–284. Springer, 2001.

[32] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, volume 6302 of *Lecture Notes in Computer Science*, pages 504–517. Springer, 2010.

[33] Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. *Computational Complexity*, 27(3):375–462, 2018.

[34] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633. ACM, 2014.

[35] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum algorithm for the Hamiltonian NAND tree. *Theory of Computing*, 4(1):169–190, 2008.

[36] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013.

[37] Michael A. Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 946–955, 2018.

[38] Sergey B. Gashkov and Igor S. Sergeev. Complexity of computation in finite fields. *Journal of Mathematical Sciences*, 191(5):661–685, 2013.

[39] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[40] 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, July 9-12, 2019, Patras, Greece*, pages 66:1–66:15, 2019.

[41] Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Computational Complexity*, 27(2):245–304, 2018.

[42] Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

[43] Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM J. Comput.*, 47(2):493–523, 2018.

[44] Johan Håstad. Almost optimal lower bounds for small depth circuits. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20. ACM, 1986.

[45] Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

[46] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[47] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 247–258. IEEE Computer Society, 2018.

[48] Shuichi Hirahara and Rahul Santhanam. On the average-case complexity of MCSP and its variants. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[49] Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as oracle. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[50] John M. Hitchcock and Aduri Pavan. On the NP-completeness of the Minimum Circuit Size Problem. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 236–245. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[51] Peter Høyer, Troy Lee, and Robert Spalek. Negative weights make adversaries stronger. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 526–535. ACM, 2007.

[52] 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, January 12-14, 2020, Seattle, Washington, USA*, pages 34:1–34:26, 2020.

[53] Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 22:1–22:36. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[54] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Hardness on any samplable distribution suffices: New characterizations of one-way functions by meta-complexity. *Electron. Colloquium Comput. Complex.*, 28:82, 2021.

[55] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995.

[56] Russell Impagliazzo and Leonid A. Levin. No better ways to generate hard NP instances than picking uniformly at random. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume II*, pages 812–821. IEEE Computer Society, 1990.

[57] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):11:1–11:16, 2019.

[58] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptol.*, 9(4):199–216, 1996.

[59] Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993.

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

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

[62] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 73–79. ACM, 2000.

[63] 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 Shubhangi Saraf, editor, *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 15:1–15:41. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[64] Bala Kalyanasundaram and Georg Schnitger. Communication complexity and lower bounds for sequential computation. In *Informatik, Festschrift zum 60. Geburtstag von Günter Hotz*, pages 253–268. Teubner / Springer, 1992.

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

[66] Sophie Laplante, Troy Lee, and Mario Szegedy. The quantum adversary method and classical formula size lower bounds. *Computational Complexity*, 15(2):163–196, 2006.

[67] Chin Ho Lee. Fourier bounds and pseudorandom generators for product tests. In *Proceedings of the 34th Computational Complexity Conference (CCC)*, pages 7:1–7:25, 2019.

[68] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

[69] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, Third Edition*. Texts in Computer Science. Springer, 2008.

[70] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020.

[71] Yanyi Liu and Rafael Pass. Cryptography from sublinear-time average-case hardness of time-bounded kolmogorov complexity. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 722–735. ACM, 2021.

[72] Yanyi Liu and Rafael Pass. A note on one-way functions and sparse languages. *IACR Cryptol. ePrint Arch.*, 2021:890, 2021.

[73] Yanyi Liu and Rafael Pass. On one-way functions from NP-complete problems. *Electron. Colloquium Comput. Complex.*, 28:59, 2021.

[74] Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on $\mathsf{EXP} \neq \mathsf{BPP}$. *Electron. Colloquium Comput. Complex.*, 28:56, 2021.

[75] Shachar Lovett and Srikanth Srinivasan. Correlation bounds for poly-size AC^0 circuits with $n^{1-o(1)}$ symmetric gates. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José

D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, volume 6845 of *Lecture Notes in Computer Science*, pages 640–651. Springer, 2011.

[76] Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Israel Symposium on Theory of Computing Systems (ISTCS)*, pages 18–24, 1993.

[77] Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 1215–1225, 2019.

[78] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[79] Cody D. Murray and Richard Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *Proceedings of the 30th Conference on Computational Complexity (CCC)*, pages 365–380, 2015.

[80] Mikito Nanashima. On basing auxiliary-input cryptography on NP-hardness via non-adaptive black-box reductions. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 185 of *LIPIcs*, pages 29:1–29:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[81] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 213–223, 1990.

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

[83] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[84] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1):43–52, 1996.

[85] Ryan O'Donnell, Rocco A. Servedio, and Li-Yang Tan. Fooling polytopes. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 614–625. ACM, 2019.

[86] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 27:1–27:29, 2019.

[87] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 18:1–18:49. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[88] Igor Carboni Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer*

Science, FOCS 2018, Paris, France, October 7-9, 2018, pages 65–76. IEEE Computer Society, 2018.

[89] Mike Paterson and Uri Zwick. Shrinkage of de Morgan formulae under restriction. *Random Struct. Algorithms*, 4(2):135–150, 1993.

[90] Ján Pich. Learning algorithms from circuit lower bounds. *CoRR*, abs/2012.14095, 2020.

[91] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[92] Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Inf.*, 25(5):515–535, 1988.

[93] Alexander A. Razborov and Steven Rudich. Natural proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 204–213, 1994.

[94] Ben Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every boolean function. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 544–551. IEEE Computer Society, 2009.

[95] Ben Reichardt. Reflections for quantum query algorithms. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 560–569. SIAM, 2011.

[96] Ben Reichardt and Robert Spalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(1):291–319, 2012.

[97] Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. *Electron. Colloquium Comput. Complex.*, 28:57, 2021.

[98] Rahul Santhanam. Pseudorandomness and the Minimum Circuit Size Problem. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *LIPIcs*, pages 68:1–68:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[99] Rocco A. Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 30:1–30:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[100] Rocco A. Servedio and Li-Yang Tan. Luby-velickovic-wigderson revisited: Improved correlation bounds and pseudorandom generators for depth-two circuits. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 56:1–56:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[101] Rocco A. Servedio and Li-Yang Tan. Improved pseudorandom generators from pseudorandom multi-switching lemmas. In Dimitris Achlioptas and László A. Végh, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*, volume 145 of *LIPIcs*, pages 45:1–45:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[102] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28:59–98, 1949.

[103] Bella Abramovna Subbotovskaya. Realization of linear functions by formulas using $\vee$, &, $-$. In *Doklady Akademii Nauk*, volume 136, pages 553–555. Russian Academy of Sciences, 1961.

[104] Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 551–560. IEEE Computer Society, 2014.

[105] Avishay Tal. #SAT algorithms from shrinkage. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:114, 2015.

[106] Avishay Tal. Formula lower bounds via the quantum method. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1256–1268. ACM, 2017.

[107] Avishay Tal. Tight bounds on the fourier spectrum of AC0. In Ryan O'Donnell, editor, *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, volume 79 of *LIPIcs*, pages 15:1–15:31. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[108] Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

[109] Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC0. In *Proceedings of the 28th Conference on Computational*

Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013, pages 242–247. IEEE Computer Society, 2013.

[110] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

[111] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[112] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM J. Comput.*, 36(5):1387–1403, 2007.

[113] Emanuele Viola. Pseudorandom bits and lower bounds for randomized Turing machines. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:51, 2019.

[114] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013.

[115] Osamu Watanabe. The time-precision tradeoff problem on on-line probabilistic Turing machines. *Theor. Comput. Sci.*, 24:105–117, 1983.

[116] Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987.

[117] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 80–91. IEEE Computer Society, 1982.

Appendix A

Omitted Proofs

A.1 Hard-on-average Problems in NP

We first introduce some useful notation. For a language $L \subseteq \{0,1\}^*$ we define its *characteristic function*, namely $f_L : \{0,1\}^* \to \{0,1\}$, to be a function given by

$$f_L(x) := \begin{cases} 1 & \text{if } x \in L, \\ 0 & \text{otherwise} \end{cases}$$

for all $x \in \{0,1\}^*$.

For sets $K, L \subseteq \{0,1\}^*$, the *disjoint union of K and L*, denoted $K \uplus L$, is the set $\{0x \mid x \in K\} \cup \{1x \mid x \in L\}$.

For a failure parameter function $\alpha : \mathbb{N} \to [0,1]$, we say that a language L is *α-hard-on-average (α-HoA)* if its characteristic function f_L is α-HoA. Similarly, we define average-case easiness for languages.

We prove the following.

Proposition A.1. *Let L be a language in* NP *that is α-HoA for some failure parameter function $\alpha : \mathbb{N} \to [0,1]$. Then, the language $L^* := L \uplus \mathrm{SAT}$ is* NP*-complete and α^*-HoA, where $\alpha^* : \mathbb{N} \to [0,1]$ is a failure parameter function such that $\alpha^*(n) := \alpha(n-1) - 1/2$ for all naturals $n \geq 2$.*

Before we prove Proposition A.1, we recount the following basic observation.

Lemma A.2. NP *is closed under disjoint union.*

We now turn to the proof of Proposition A.1.

Proof of Proposition A.1. By Lemma A.2, the language L^* is in NP since L^* is the disjoint union of $L \in \mathsf{NP}$ and $\mathrm{SAT} \in \mathsf{NP}$.

We will now show that L^* is NP-hard, by giving a polynomial-time reduction R from SAT to L^*. For all $x \in \{0,1\}^*$, let $R(x) := 1x \in \{0,1\}^*$. We see that R is polynomial-time computable. Moreover, if $x \in \mathrm{SAT}$, then $R(x) = 1x \in L^*$, and if $R(x) \in L^*$, then $1x \in L^*$ and so $x \in \mathrm{SAT}$.

What is left is to prove that L^* is α^*-HoA, where $\alpha^* : \mathbb{N} \to [0,1]$ is such that $\alpha^*(n) := \alpha(n-1) - 1/2$ for all naturals $n \geq 2$. Towards a contradiction, assume that L^* is $(1-\alpha^*)$-EoA and let H^* be a heuristic that witnesses this phenomenon. We will give a heuristic H that witnesses the fact that L is $(1-\alpha)$-EoA, whereby establishing the desired contradiction. To this end, let

$$H(x) := H^*(0x)$$

for all $x \in \{0,1\}^*$. We will show that H has the desired average-case performance. That is,

$$\begin{aligned} \mathop{\mathbf{Pr}}_{x \sim \{0,1\}^n}[H(x) = f_L(x)] &= \mathop{\mathbf{Pr}}_{x \sim \{0,1\}^n}[H^*(0x) = f_{L^*}(0x)] \\ &= \mathop{\mathbf{Pr}}_{y \sim \{0,1\}^{n+1}}[H^*(y) = f_{L^*}(y) \mid y_1 = 0] \end{aligned}$$

$$
\begin{aligned}
&\geq \mathop{\mathbf{Pr}}_{y\sim\{0,1\}^{n+1}}[H^*(y) = f_{L^*}(y)] - \mathop{\mathbf{Pr}}_{y\sim\{0,1\}^{n+1}}[y_1 = 1] \\
&\geq 1 - \alpha^*(n+1) - \frac{1}{2} \\
&= 1 - \left(\alpha((n+1)-1) - \frac{1}{2}\right) - \frac{1}{2} \\
&= 1 - \alpha(n)\,.
\end{aligned}
$$

□

Index