# Worst-Case and Average-Case Complexity of Minimum Circuit Size Problem and Its Variants
# 回路最小化問題とその変種の最悪時及び平均時計算量

by

Shuichi Hirahara
平原 秀一

A Doctor Thesis
博士論文

Submitted to
the Graduate School of the University of Tokyo
on December 7, 2018
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Information Science and Technology
in Computer Science

Thesis Supervisor: Hiroshi Imai　今井 浩
Professor of Computer Science

## ABSTRACT

Although public-key cryptography is a fundamental and extensively used technology, its security relies on unproven assumptions much stronger than $\mathsf{P} \neq \mathsf{NP}$. One of the ultimate goals of complexity theory is to establish secure public-key cryptography; however, there is a long way to go. Impagliazzo (1995) proposed five possible worlds, Algorithmica, Heuristica, Pessiland, Minicrypt, Cryptomania, which are consistent with our current knowledge of complexity theory. These five worlds are classified according to the truth of the following central open questions: $\mathsf{P} \neq \mathsf{NP}, \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$, the existence of a one-way function, and the existence of public-key cryptography. The open problems are listed in increasing order of the strength of the hypothesis, and the converse directions are important open problems in complexity theory; that is, True $\stackrel{?}{\Rightarrow}$ $\mathsf{P} \neq \mathsf{NP}$ $\stackrel{?}{\Rightarrow}$ $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ $\stackrel{?}{\Rightarrow}$ $\exists$ one-way functions $\stackrel{?}{\Rightarrow}$ $\exists$ public-key cryptography. By establishing one implication, one possible world is excluded from Impagliazzo's five worlds. And if the four implications are proved, it is concluded that our world is Cryptomania, i.e., computationally-secure public-key cryptography exists.

There are significant obstacles to resolving these open questions. For example, any proof showing that $\mathsf{P} \neq \mathsf{NP} \stackrel{?}{\Rightarrow} \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ (which corresponds to excluding Heuristica) must be a *non-relativizing* proof technique, and must overcome the limits of *black-box reductions* unless the polynomial-time hierarchy collapses.

In this thesis, we focus on the central problems in complexity theory called *Minimum Description Length Problems* (MDLPs), and overcome one of the aforementioned obstacles. Minimum Description Length Problems ask the minimum length of a "program" that compresses a given string. One instantiation of minimum description length problems is called the *Minimum Circuit Size Problem* (MCSP), which asks for compressing a given string by a truth table of a circuit. Another instantiation is called the *Minimum Time-bounded Kolmogorov Complexity Problem* (MINKT or MKTP), which asks for compressing a given string by an efficient Turing machine. We establish the equivalence between the worst-case and average-case complexity of an approximation version of these problems. This is shown by a non-black-box reduction technique, and under some plausible assumptions, our results overcome the limits of black-box reductions. Our results can be seen as a new approach towards excluding Heuristica: In order to establish the equivalence between the worst-case and average-case complexity of $\mathsf{NP}$, it suffices to prove the $\mathsf{NP}$-hardness of approximation versions of minimum description length problems.

Next, we present various hardness results of MCSP. We show that MDLPs are hard under average-case complexity conjectures such as the Planted Clique Conjecture and Random 3SAT, thereby strengthening evidence that MCSP $\notin \mathsf{coNP}$. We also show that every auxiliary-input one-way function can be inverted with MCSP oracles, which improves the $\mathsf{SZK}$ (statistical zero knowledge) hardness of MCSP shown by Allender and Das (2017).

Then, as with the limits of black-box reductions, we show that current reduction techniques have a certain limit that prevents us from improving the $\mathsf{SZK}$-hardness to $\mathsf{NP}$-hardness of MCSP: Specifically, we introduce the notion of *oracle-independent reductions*, observe that most of the current reductions to MCSP are oracle-independent, and show that an oracle-independent reduction technique is not likely to be used to establish $\mathsf{NP}$-hardness of MCSP.

In light of this barrier, we turn our attention to MCSP for restricted circuit classes. Masek (1979) proved the $\mathsf{NP}$-hardness of MCSP for $\mathsf{DNF}$ formulas, that is, the minimum circuit size problem for depth-2 circuits. It was open for nearly four decades to obtain $\mathsf{NP}$-hardness of MCSP for any circuit class more expressive than $\mathsf{DNF}$ formulas, despite the fact that the question is recognized as important. We resolve this open question by establishing $\mathsf{NP}$-hardness of MCSP for $\mathsf{DNF} \circ \mathsf{XOR}$ circuits.

We also present several additional results including: Strong evidence against Allender's conjecture (2012), which roughly states that a computability-theoretic analogue of MCSP is not $\mathsf{NP}$-hard under polynomial-time nonadaptive reductions; The first natural $\mathsf{NP}$-intermediate problems supported by weak complexity-theoretic assumptions; Improved connections between hardness of MCSP and circuit lower bounds; Unconditional

circuit lower bounds for MCSP.

## 論文要旨

今日の情報通信社会の基盤技術として公開鍵暗号系が広く使われているが、その安全性は $\mathsf{P} \neq \mathsf{NP}$ よりもはるかに強い仮定に基づいており未解決である。計算量理論の究極的な目標のひとつは数学的に証明された公開鍵暗号系を構成することにあるが、計算量理論には $\mathsf{P} \neq \mathsf{NP}$ を始めとして未解決問題が山積している。Impagliazzo (1995) の分類に従うと、我々の現在の計算量理論の知識と一貫性のある 5 つの世界 (Algorithmica, Heuristica, Pessiland, Minicrypt, Cryptomania) がある。これらは $\mathsf{P} \neq \mathsf{NP}, \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ 一方向性関数の存在, 公開鍵暗号の存在の 4 つの未解決問題が成立するか否かによって分類されており、どれか一つの世界が真の世界に対応している。これらの未解決問題は仮定の強い順に並んでおり、その逆を示すことは計算量理論における中心的な未解決問題である。つまり、True $\overset{?}{\Rightarrow}$ $\mathsf{P} \neq \mathsf{NP}$ $\overset{?}{\Rightarrow}$ $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ $\overset{?}{\Rightarrow}$ $\exists$ 一方向性関数 $\overset{?}{\Rightarrow}$ $\exists$ 公開鍵暗号。ひとつの矢印「$\overset{?}{\Rightarrow}$」を示すことはひとつのありうる世界を除外することに対応し、4 つのすべての矢印を示すことによって安全な公開鍵暗号系の存在を示すことができる。

しかしながらそれら中心的な問題の解決には重大な障壁があることが知られている。例えば $\mathsf{P} \neq \mathsf{NP} \overset{?}{\Rightarrow} \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ を解決する（つまり、Heuristica を除外する）ためには、**相対化する**証明技法では示せないし、多項式階層が潰れない限り**ブラックボックス帰着**の限界を突破する必要がある。

本論文では計算量理論における中心的な問題である**最小記述量問題**に着目し、その障壁のひとつを突破する。最小記述量問題とは、与えられた文字列を最小の「プログラム」へと圧縮したときの長さを問う問題である。プログラムが回路で表現される場合には回路最小化問題 (Minimum Circuit Size Problem; MCSP) と呼ばれ、プログラムが効率的なチューリングマシンで表現される場合には最小時間制限付きコルモゴロフ記述量問題 (MINKT または MKTP) と呼ばれる。これらの近似問題に関して、平均時計算量と最悪時計算量が同値になることを示す。これはブラックボックスでない帰着手法で証明されており、適切な仮定の下でブラックボックス帰着を突破している初めての結果である。この結果は Heuristica を除外するための新しいアプローチとしても見ることができる。すなわち、NP の最悪時・平均時計算量の同値性を示すためには、近似版最小記述量問題の NP 完全性を解決すれば十分である。

次に、MCSP の様々な困難性を示す。埋め込みクリーク予想やランダム 3SAT などに関する平均計算量の仮定の下で MCSP や MKTP が計算困難であることを示すことにより、$\mathrm{MCSP} \notin \mathsf{coNP}$ であるというさらなる証拠を与える。また、MCSP オラクルの下で補助入力一方向性関数を破れることを示し、Allender と Das (2017) による MCSP の統計的ゼロ知識証明 (SZK) 困難性を改善する。

ブラックボックス帰着の限界と同様に、SZK 困難性を NP 困難性に改善するためには現在の証明手法には限界があることを示す。具体的には**オラクル独立帰着**という概念を導入し、ほとんどの MCSP への帰着手法はオラクル独立であり、そのような手法では NP 完全性を解決することができないという証拠を示す。

それゆえ、(一般の回路最小化問題ではなく) 制限された回路クラスに関する回路最小化問題に着目する。Masek (1979) により DNF に対する回路最小化問題は NP 完全だと知られていたが、DNF よりも表現能力の高い回路クラスに関する回路最小化問題についての NP 完全性は (重要性が認識されていたにも関わらず) 約 40 年間未解決であった。本論文

では DNF ∘ XOR 回路最小化問題の NP 完全性を解決する。

さらにいくつかの追加の結果を示す。大雑把に言うと「計算可能性版の MCSP が非適応多項式時間帰着のもとで NP 困難ではない」という Allender の予想 (2012) を否定する証拠を示す。弱い計算量理論の仮定に基づく初めての自然な NP の中間の問題を構成する。MCSP と回路下界の関係を改善する。仮定なしでの MCSP に対する回路下界を示す。

# Acknowledgements

I started doing research on complexity theory as an undergraduate student when Akitoshi Kawamura suggested working on Allender's conjecture about the efficient reductions to Kolmogorov complexity. I was fortunate enough to present some progress about the conjecture at MFCS'14 [HK18], where Eric Allender was also there to present his paper with Bireswar Das about MCSP [AD17]. In addition to the fact that his paper won the Best Paper Award of the conference, the presentation was quite nice enough to attract my interest to MCSP. Then I started some work on the new notion called selector. I did not know how to write a paper at that time, but thanks to the generous guidance of my advisor, Prof. Hiroshi Imai, the paper was accepted to CCC'15 [Hir15]. I was again fortunate enough to attend the talk of Cody Murray, who presented his paper [MW17] with Ryan Williams about MCSP, which gave me an impression that I could do something about it.

Going back to Japan, during the seminar of ELC (Exploring the Limitations of Computation), Osamu Watanabe and I started working on MCSP. I had a number of enjoyable discussions with him, and we were able to come up with the notion of oracle-independent reductions, and the paper was accepted to CCC'16 [HW16]. At the conference, I met Rahul Santhanam, who kindly invited me to visit Oxford university.

Before going to Oxford university, supported by Grant-in-Aid for JSPS Research Fellow, I visited Eric Allender of Rutgers university. Eric is patient and generous enough to discuss with me many times, and I had an enjoyable collaboration with him [AH17].

Then I visited Oxford university, and I worked with Rahul, the results of which appeared in [HS17]. He was very intelligent and had a lot of ideas, and he was kind enough to share these with me. I do feel that my cleaner view on MCSP was shaped during the collaboration with Rahul. In particular, the view about "meta-complexity" was due to him. Without meeting him and collaborating with him, I am quite certain that I could not have been able to contribute complexity theory.

My view of complexity theory was also sharpened through presentations for non-specialists. I was fortunate to have opportunities to give presentations at ELC, JST ERATO Kawarabayashi Large Graph Project, and ACT-I, JST.

Thanks to the grant from ACT-I, I was able to visit Oxford again, and there I worked with Rahul Santhanam and Igor C. Oliveira [HOS18], and it was a quite productive and enjoyable visit.

Without collaborating with Osamu Watanabe, Eric Allender, Rahul Santhanam, and Igor C. Oliveira and without precious advice and encouragement of Prof. Hiroshi Imai, I could not have been able to achieve what I achieved. I would like to express my deep gratitude to all the people that helped me in any aspect of the research. Now that I do have much clearer understanding about MCSP and related problems, I was able to essentially disprove Allender's con-

jecture in this thesis. I do feel that I am quite lucky to meet the wonderful co-authors, the advisor, my friends, and my parents.

I am also very honored to have Prof. Naoki Kobayashi as the chair, and Prof. Akitoshi Kawamura, Prof. Masami Hagiya, Prof. Tetsuo Shibuya, Prof. Noboru Kunihiro, and Prof. Junya Honda as the members of the committee for this thesis. I highly appreciate them taking the time and effort to study and examine this thesis.

There is no enough space to mention all of them, but I do thank all the people that interacted and supported me in any aspect of my life.

# Contents

# Chapter 1

# Introduction

*Computational Complexity Theory* aims at understanding what computational problems can or cannot be solved efficiently by an abstract model of computational devices, such as Turing machines or logic circuits. The main focus of computational complexity is to prove limits of computation. One of the most central open questions is the $\mathsf{P} \stackrel{?}{\neq} \mathsf{NP}$ problem, which intuitively asks whether verifying the correctness of a proof efficiently is harder than finding a proof efficiently. Not only mathematically challenging and interesting is to prove such a *nonexistence* of an efficient algorithm, it is also the theory that supports modern cryptography widely used to protect secret information in our daily life.

Public-key cryptography is a fundamental technology that enables us to communicate securely without revealing secret information to an eavesdropper. Despite the fact that public-key cryptography is widely used, no public-key cryptography is proved to be secure.

For example, one of the most commonly used public-key cryptosystems is RSA, developed by Rivest, Shamir, and Adleman [RSA78]. The cryptosystem is based on intractability of integer factorization, and RSA is secure only if one cannot solve integer factorization efficiently. It should be noted that integer factorization is a problem in $\mathsf{NP} \cap \mathsf{coNP}$, and thus there is not so strong complexity-theoretic evidence supporting why integer factorization should be intractable. In the breakthrough work of Shor [Sho99], it was shown that a quantum computer can solve integer factorization in polynomial time; thus RSA is not secure against quantum computers. Because of Shor's quantum algorithm, there have been a lot of work done in order to construct post-quantum cryptography – that is, cryptography secure even against quantum computers.

Even worse, if $\mathsf{P} = \mathsf{NP}$, then essentially all cryptographic primitives can be broken by a *classical computer*. This is intuitively because $\mathsf{NP}$ is the complexity class of problems whose YES instances have a certificate that is efficiently checkable; in order to build a meaningful cryptographic primitive, it is required that a legitimate user who has a secret key (i.e., a certificate) must be verified efficiently. More formally, the existence of a *one-way function* (OWF) is often considered as a minimal complexity assumption to build cryptography [IL89]. Roughly speaking, a one-way function is a function $f$ such that $f$ is easy to compute but no efficient adversary can invert $f$ on average; it is easy to see that any one-way function can be inverted with $\mathsf{NP}$ oracles. Thus the security of cryptographic primitives must be based on intractability of some $\mathsf{NP}$ problem, and hence proving $\mathsf{P} \neq \mathsf{NP}$ is the first step towards the construction of secure cryptography.

However, there is a huge gap between $\mathsf{P} \neq \mathsf{NP}$ and public-key cryptography. This gap was already discussed in the seminal work of Diffie and Hellman [DH76], which introduced public-key cryptography. The main problem is that traditional

complexity classes such as $\mathsf{P}$ and $\mathsf{NP}$ measure the performance of an algorithm with respect to the *worst-case* input; therefore, the statement $\mathsf{P} \neq \mathsf{NP}$ only tells us that there exists *some* hard input on which a polynomial-time machine cannot solve some $\mathsf{NP}$ problem; however, it does not tell us how to generate such a hard input efficiently. In contrast, for the purpose of cryptography, we need to efficiently generate a secret key randomly so that an adversary cannot find the secret key in a reasonable amount of time. Thus we need to understand the *average-case complexity* of $\mathsf{NP}$: that is, how much time on average does it take to compute $\mathsf{NP}$ problems on efficiently generated random inputs?

### 1.1 Impagliazzo's Five Worlds

In the very influential survey on average-case complexity of Impagliazzo [Imp95], the gap between $\mathsf{P} \neq \mathsf{NP}$, average-case complexity of $\mathsf{NP}$, and the existence of cryptographic primitives was clearly addressed. He explored five possible scenarios that are consistent with our current knowledge on complexity theory, and named each possible world as follows: Algorithmica (where $\mathsf{NP}$ is easy on the worst-case; e.g. $\mathsf{P} = \mathsf{NP}$), Heuristica (where $\mathsf{NP}$ is hard on the worst-case, but easy on the average-case; e.g. $\mathsf{P} \neq \mathsf{NP}$ and $\mathsf{DistNP} \subseteq \mathsf{AvgP}$), Pessiland (where $\mathsf{NP}$ is hard on average, but there is no one-way function), Minicrypt (where a one-way function exists, but no public-key cryptography exists), and Cryptomania (public-key cryptography exists). Most of us believe that we live in Cryptomania, but we do not know the truth yet. The five worlds are classified according to the four central open questions in complexity theory, and exactly one of the possible worlds corresponds to our world.

What is known about Impagliazzo's five worlds? The list of the five worlds is known to be in "decreasing order" of the power of polynomial-time machines; that is, $\exists$ public-key cryptography $\Rightarrow$ $\exists$ one-way functions $\Rightarrow$ $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ $\Rightarrow$ $\mathsf{P} \neq \mathsf{NP}$. The converse directions of these implications are important open questions in complexity theory; that is, True $\stackrel{?}{\Rightarrow}$ $\mathsf{P} \neq \mathsf{NP}$ $\stackrel{?}{\Rightarrow}$ $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$ $\stackrel{?}{\Rightarrow}$ $\exists$ one-way functions $\stackrel{?}{\Rightarrow}$ $\exists$ public-key cryptography. By establishing one implication, one possible world is excluded from Impagliazzo's five worlds. And if the four implications are proved, it is concluded that our world is Cryptomania, i.e., computationally-secure public-key cryptography exists.

Since all these questions are the central open questions in complexity theory and cryptography, a lot of work has been done for each open question in order to understand why current proof techniques are not likely to resolve the questions. For example, in order to resolve $\mathsf{P} \neq \mathsf{NP}$ (or, in other words, to exclude Algorithmica from the possible worlds), it is known that one needs to develop a new proof technique that overcomes the *relativization* barrier [BGS75], the *algebrization* barrier [AW09], and the *natural proof* barrier [RR97] simultaneously. It is known that $\mathsf{P} \neq \mathsf{EXP}$ by the time hierarchy theorem [HS65], but such a proof technique is known to be relativizing, i.e., $\mathsf{P}^A \neq \mathsf{EXP}^A$ for every oracle $A$. However, Baker, Gill, and Solovay [BGS75] showed that $\mathsf{P}^A = \mathsf{NP}^A$ for some oracle $A$ (and moreover $\mathsf{P}^B \neq \mathsf{NP}^B$ for some oracle $B$), and thus a relativizing proof technique is not enough to resolve $\mathsf{P} \neq \mathsf{NP}$. One of the main non-relativizing proof techniques in complexity theory, called *algebrization*, is developed through the study of interactive proof systems and the PCP theorem [ALM+98], but it turned out that the proof technique also has a limit as shown by Aaronson and Wigderson [AW09]. In 1980s, combinatorial approaches for showing limits of constant depth circuits have been quite successful, culminating in exponen-

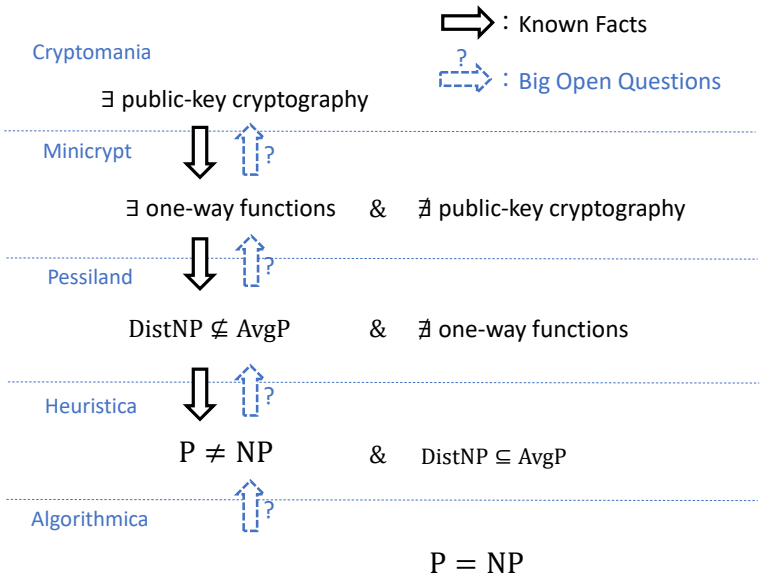


Figure 1.1: Impagliazzo's Five Worlds and Central Open Questions.

tial circuit size lower bounds for $\mathsf{AC}^0$ circuits [Hås86] and $\mathsf{AC}^0[p]$ circuits for a prime $p$ [Raz87, Smo87]. It turned out that such an approach has a limit as well: Razborov and Rudich [RR97] proposed a *natural proof* barrier, and showed that "natural" proof techniques are not likely to prove a lower bound for $\mathsf{TC}^0$, i.e, the class of constant-depth circuits with threshold gates.

Similarly, in order to exclude Heuristica (e.g., $\mathsf{P} \neq \mathsf{NP} \implies \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$), one needs to develop a new proof technique that overcomes a relativization barrier (as shown by Impagliazzo [Imp11]) and *limits of black-box reductions*. One of the most prevailing proof techniques for showing intractability of a problem in complexity theory is by means of *reductions*, that is, we compare hardness of two different computational tasks by reducing one task to another. However, (black-box) reductions have certain limits and are not likely to be able to exclude Heuristica, as shown by Feigenbaum and Fortnow [FF93] and Bogdanov and Trevisan [BT06b]. More specifically, any nonadaptive black-box reduction technique cannot reduce any worst-case problem outside $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ to an average-case problem in $\mathsf{NP}$.

At this point, a natural question is how to tackle the challenging open questions. The main theme of this thesis is that *meta-computational problems* asking for compression of a given string might be a key to overcoming these barriers. Here "meta-complexity" refers to the complexity of problems that encode questions about algorithms or computations. Historically, complexity theory was advanced together with improved understanding of meta-computational problems. A canonical example of meta-computational problems is the satisfiability problem ($\mathsf{SAT}$): Given as input a representation of an algorithm $A$ in a certain form such as 3CNF formulas or Boolean circuits, the task is to decide whether there is an input on which the algorithm $A$ outputs "YES". $\mathsf{SAT}$ has had major impacts on complexity theory: For example, $\mathsf{SAT}$ was one of the first problems identified as an $\mathsf{NP}$-complete problem by Cook and Levin [Coo71, Lev73]; the celebrated PCP theorem shows inapproximability of $\mathsf{SAT}$ [ALM+98]; more recently, a connection between circuit lower bounds and non-trivial $\mathsf{SAT}$ algorithms was established by Williams [Wil13, Wil14], whose proof technique is an important candidate that overcomes the natural proof barrier.

## 1.2 Minimum Circuit Size Problem (MCSP) and Its Variants

Another meta-computational problem in some sense dual to SAT is the Minimum Circuit Size Problem (MCSP): Given the entire truth table of a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ and a size parameter $s \in \mathbb{N}$, the task is to decide whether $f$ can be computed by a circuit of size at most $s$. While SAT asks a property of a Boolean function succinctly encoded by an algorithm, MCSP asks whether a Boolean function represented in a lengthy way can be compressed into a small circuit. It is easy to see that MCSP $\in$ NP, but its precise complexity is not well understood. This is despite the fact that MCSP has attracted researchers of the Soviet Union as early as 1950s, as surveyed in an informative account of Trakhtenbrot [Tra84]. It is also reported in [AKRR11] that Levin delayed his publication of SAT [Lev73] because he hoped to show NP-completeness of MCSP. Nearly half century going to pass, no one has found an NP-completeness proof, nor has any strong evidence against NP-completeness of MCSP been found.

Problems of minimizing time-bounded Kolmogorov complexity are often regarded as variants of MCSP. MINKT (Minimum Kolmogorov Time-bounded Complexity [Ko91]) asks the minimum *program size* to output a given string $x$ within a given time bound $t$: specifically, given a string $x$ and integers $t, s$ represented in unary, it asks whether there is a program of size $\leq s$ that outputs $x$ within $t$ steps. Another variant called MKTP [ABK$^+$06b, AHK17] asks for minimizing $s + t$, i.e., the program size plus the time it takes to output $x$ by a random access machine. It was shown by Ko [Ko91] that a relativizing proof technique is not enough to resolve NP-completeness of MINKT.

Common to these problems is to ask for *compression by algorithms*: Given a string $x$, the task is to check whether $x$ can be compressed into a short description of an algorithm that outputs $x$. In short, MCSP is the problem of compressing a given input by the truth table of a circuit; MINKT and MKTP is the problem of compressing a given input by an efficient Turing machine. It is often the case that it is easier to compress a string by a program than a truth table of a circuit; thus every known hardness result for MCSP works for MKTP and MINKT, but there are several results for MKTP and MINKT that are not known to hold for MCSP. We call these problems as Minimum Description Length Problems (MDLPs).

Why do we think that studying MCSP and its variants is valuable? The reason is that MCSP is arguably one of the central meta-computational problems in complexity theory. We show in Figure 1.2 that the problem lies in a central position of Impagliazzo's five worlds.

The fundamental relationship between MCSP and cryptography was found in the natural proof framework of Razborov and Rudich [RR97]. Roughly speaking, a *natural property* (useful against the general circuit class) is an efficient algorithm that solves MCSP well on average. It was shown in [RR97] that such an algorithm is able to break every pseudorandom function generator (PRFG) constructed by Goldreich, Goldwasser and Micali [GGM86] based on any pseudorandom generator (PRG). Håstad, Impagliazzo, Levin and Luby [HILL99] constructed a pseudorandom generator from any one-way function. Combining these constructions, the relationship can be stated as the implication "$\exists$ one-way functions $\Longrightarrow$ $\exists$ MCSP $\notin$ BPP".

Dual to cryptography is *learning theory*. While cryptography aims at constructing an efficiently computable function $f$ that does not reveal any secret, learning theory aims at learning such a function $f$ by a small circuit. A connection between MCSP and learning theory was clearly established by Carmosino, Impagliazzo, Kabanets, and Kolokolova [CIKK16, CIKK17]. They showed that

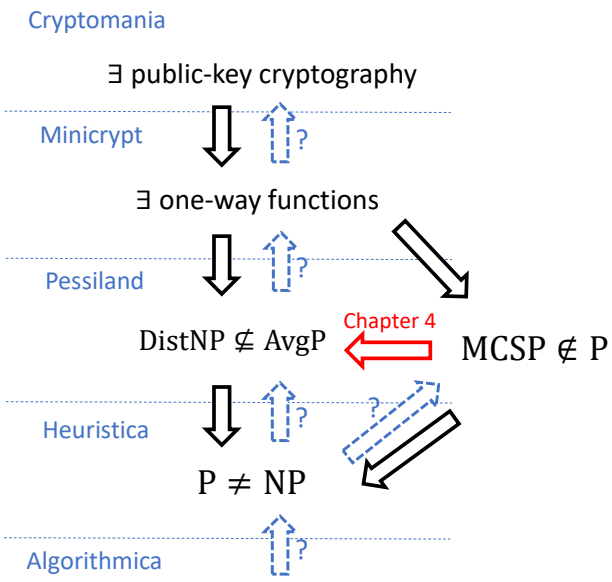


Figure 1.2: Impagliazzo's five worlds, MCSP, and our results. Here we ignore the details such as P and BPP and MCSP and its approximation version.

learning small circuits under the uniform distribution can be done with oracle access to any natural property (and in particular MCSP), by breaking a complexity-theoretic pseudorandom generator. In particular, they were able to establish the first quasi-polynomial-time learning algorithm for $\mathsf{AC}^0[p]$, because the circuit lower bound proofs of Razborov [Raz87] and Smolensky [Smo87] yield a natural property useful against $\mathsf{AC}^0[p]$ circuits.

Based on the natural proof framework, Kabanets and Cai [KC00] named the problem as MCSP and reawoke interest in MCSP. Since then many efforts have been made to understand the complexity of MCSP (e.g., [ABK+06b, AHM+08, AKRR11, AD17, MW17, HP15, HW16, CIKK16, OS17, HS17, AH17, AGvM+18, IKV18, HOS18, Hir18]). However, it is still a long-standing open question whether MCSP is NP-hard. The open question is depicted in Figure 1.2 as the implication "$\mathsf{NP} \neq \mathsf{P} \stackrel{?}{\Rightarrow}$ MCSP $\notin \mathsf{P}$." (Recall that a problem $L$ is NP-hard under polynomial-time Turing reductions if $\mathsf{NP} \subseteq \mathsf{P}^L$. We can state it equivalently as $\mathsf{NP} \not\subseteq \mathsf{P}^R \Rightarrow L \notin \mathsf{P}^R$ for every oracle $R$. The unrelativized implication $\mathsf{NP} \not\subseteq \mathsf{P} \Rightarrow L \notin \mathsf{P}$ gives rise to the weakest notion of NP-hardness.)

## 1.3 Overcoming Limits of Black-Box Reductions

The main contribution of this thesis is to prove the first *non-black-box* worst-case to average-case reductions: We prove that if MCSP is hard in the worst-case sense, then its average-case version is also hard (depicted in Figure 1.2 as "MCSP $\notin \mathsf{P} \implies \mathsf{DistNP} \not\subseteq \mathsf{AvgP}$" informally). More specifically, we show that if an approximation version $\mathrm{Gap}_\epsilon$MCSP is not in BPP then $\mathsf{DistNP} \not\subseteq \mathsf{AvgBPP}$. Here $\mathrm{Gap}_\epsilon$MCSP denotes the problem of approximating the minimum $n$-variable circuit size within a factor of $2^{(1-\epsilon)n}$ for a constant $\epsilon > 0$. (While the approximation factor is huge for MCSP, we obtain a much better approximation in the case of MINKT.) MDLPs are conjectured to be outside $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$, and thus our results overcome the limits of black-box reductions of [FF93, BT06a].

There are few problems in NP that are believed to be intractable and admit worst-case to average-case reductions. In the seminal work of Ajtai [Ajt96], he

showed that the shortest vector problem admits worst-case to average-case reductions; however, since the reduction is black-box, the problem is known to be in $\mathsf{NP} \cap \mathsf{coNP}$ [GG00, AR05]. The worst-case to average-case reduction led to a construction of lattice-based public-key cryptosystems (*e.g.* [AD97]), and these are one of candidates for post-quantum cryptography. While our results do not yield a construction of cryptographic primitives yet, the worst-case to average-case reduction for MCSP can be regarded as a significant step towards secure cryptography.

Crucial to our non-black-box reductions is the *meta-computational* property of MCSP. To give a better sense, we briefly outline the idea. Usually, we say that a computational problem $A$ *reduces* to another problem $B$ via a reduction $R$ if one can design an efficient algorithm $R$ that solves $A$, assuming that $B$ is given as oracle. The correctness of most reduction techniques can be established no matter how $B$ is complex – such a reduction is called *black-box* in the sense that $B$ is regarded as a (potentially inefficient) black-box oracle, and we just care about the input-output behavior of $B$. In contrast, in *non-black-box* reductions, we must fail to prove the correctness of the reduction when $B$ is an algorithm that takes, e.g., a super-polynomial time.

How can this happen? The point is that, in our reduction, the problem $A$ is the meta-computational problem, i.e., (an approximation version of) MCSP. The proof of the correctness of our reduction goes roughly as follows: We translate a polynomial-time algorithm $B$ into a polynomial-size circuit $B'$ by a standard transformation, and then, using the polynomial-size circuit $B'$, we try to conclude that a given truth table can be compressed by a small circuit that *incorporates* the circuit $B'$. If $B$ can be indeed implemented as a polynomial-size circuit, then we can show that the reduction can compress the input into a small circuit. On the other hand, if $B$ cannot be implemented by a small circuit, the reduction might not compress its input any longer, and thus it might fail to solve MCSP. Thus the meta-computational property of $A$ is indeed the key for overcoming the limits of black-box reductions.

While our proof techniques are still subject to the relativization barrier, the non-black-box reductions yield a new approach towards excluding Heuristica: $\mathsf{NP}$-hardness of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ implies that Heuristica does not exist, i.e., $\mathsf{NP} \not\subseteq \mathsf{BPP}$ if and only if $\mathsf{DistNP} \not\subseteq \mathsf{AvgBPP}$ (cf. Figure 1.2). We thus propose a research program of understanding Heuristica through the lens of MCSP. This thesis contributes the research program by significantly improving our understanding of MCSP and the landscape around Heuristica. Some of our results are summarized in Figure 1.3.

## 1.4 Research Lines of MCSP

We proceed to reviewing research lines of MCSP, and highlight several contributions of this thesis.

### Hardness of MCSP

Razborov and Rudich [RR97] showed that MCSP can be used to break cryptographic primitives. By inverting one-way functions using MCSP oracles, Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [ABK$^+$06b] showed various worst-case hardness results of MCSP. For example, MCSP is hard for integer factorization and the discrete logarithm problem under randomized reductions. Building on this, Allender and Das [AD17] showed $\mathsf{SZK}$-hardness of

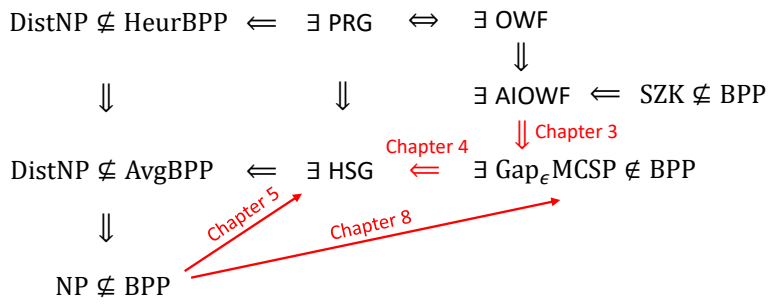


Figure 1.3: An improved landscape of Algorithmica, Heuristica, and Pessiland. "$A \to B$" means that there is no black-box reduction technique showing "$A \Rightarrow B$" under reasonable complexity theoretic assumptions. The security of all cryptographic primitives is with respect to an almost-everywhere polynomial-time randomized adversary. "$\mathsf{SZK} \not\subseteq \mathsf{BPP} \implies \exists\mathrm{AIOWF}$" is due to [Ost91], and "$\exists\mathrm{PRG} \Leftrightarrow \exists\mathrm{OWF}$" is due to [HILL99].

MCSP (where SZK stands for statistical zero knowledge), which is the current best worst-case hardness result for MCSP. The SZK-hardness provide strong evidence for intractability of MCSP with respect to polynomial-time algorithms. Unfortunately, it does not rule out the possibility that MCSP $\in \mathsf{NP} \cap \mathsf{coNP}$ because $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ [For89, AH91] and it is popular to conjecture that $\mathsf{AM} = \mathsf{NP}$ [KvM02].

In Chapter 3, we present evidence that MCSP $\notin \mathsf{coNP}$ by relying on several average-case complexity assumptions. For example, we show that MKTP is Random 3SAT-hard, thereby presenting mild evidence supporting NP-hardness of MKTP. We also show that every *auxiliary-input one-way function* (AIOWF) can be inverted by using MCSP oracles. Here an auxiliary-input one-way function is a weaker cryptographic primitive than the standard one-way function, introduced by Ostrovsky and Wigderson [OW93] in the study of zero-knowledge proofs; for example, Ostrovsky [Ost91] implicitly showed that $\mathsf{SZK} \not\subseteq \mathsf{BPP}$ implies the existence of auxiliary-input one-way functions. The implication from the existence of an auxiliary-input one-way function to MCSP $\notin \mathsf{BPP}$ (cf. Figure 1.3) was already implicit in the work of [ABK$^+$06b], but it was not explicitly shown until the work of Allender and Hirahara [AH17]. As a consequence, it gives a simple and direct proof of the SZK-hardness of MCSP. Moreover, by combining non-black-box worst-case to average-case reductions that will be presented in Chapter 4, we obtain a construction of hitting set generators (HSG) from the existence of auxiliary-input one-way functions.

### Non-Hardness of MCSP

Kabanets and Cai [KC00] explained the reason why it is difficult to prove NP-hardness of MCSP: They showed that NP-hardness of MCSP cannot be resolved by using a "natural" reduction technique unless another important open question is resolved simultaneously. For example, NP-hardness of MCSP under natural polynomial-time many-one reductions implies that there exists an exponential-time computable function that requires circuits of super-polynomial size, the latter of which is an important open question in complexity theory. (Here a reduction is called *natural* if the size parameter of the output of the reduction depends only on the input size.) The basic idea is that MCSP is intimately related to circuit lower bounds, and thus in order to prove hardness results for MCSP, we should be able to understand circuit lower bounds well. A lot of work

has been devoted to pushing this direction further (e.g., [KC00, MW17, AHK17, HP15, AH17, HW16]).

For example, Murray and Williams [MW17] removed the technical condition of the naturalness in Kabanets and Cai's result: NP-hardness of MCSP under polynomial-time many-one reductions implies $\mathsf{ZPP} \neq \mathsf{EXP}$. We will improve their results to the case of a polynomial-time Turing reduction, that is, a reduction that makes an adaptive query to an oracle, and show that NP-hardness of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ under polynomial-time Turing reductions for every $\epsilon > 0$ implies $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ in Chapter 7.

More importantly, the line of work does not explain why it is difficult to prove NP-hardness of MCSP under *randomized* reductions. It should be noted that the SZK-hardness of MCSP is proved by using randomized reductions, and hence one may wonder whether such a proof technique leads us to NP-hardness of MCSP. We show in Chapter 8 that there are inherent limits on the current reduction technique: Specifically, we introduce the notion of *oracle-independent* reductions that capture almost all current reduction techniques, and show that oracle-independent reductions are not likely to be able to reduce any problem outside $\mathsf{AM} \cap \mathsf{coAM}$ to MCSP. In particular, the SZK-hardness of MCSP is likely to be the best worst-case hardness result under our current proof techniques.

#### NP-hardness of MCSP for restricted circuit classes

Given these barriers for proving hardness results for MCSP, we consider MCSP for a restricted circuit class $\mathfrak{C}$. That is, instead of general circuits, we consider the task of finding the minimum size of $\mathfrak{C}$-circuits. Masek [Mas79] proved in 1979 that MCSP for DNF formulas is NP-hard. However, no NP-hardness result for MCSP for any circuit class more expressive than DNF formulas has been known for nearly four decades, despite the fact that the question is recognized as an important open question [AHM+08]. In Chapter 9, we resolve this open question by proving NP-hardness for $\mathsf{DNF} \circ \mathsf{XOR}$ circuits.

#### Related Work

We mention in passing several problems related to MCSP: One natural version of a circuit minimization problem is called the *Minimum Equivalent* $\mathfrak{C}$ *Expression Problem*: Given a $\mathfrak{C}$-circuit $C$, find the smallest $\mathfrak{C}$-circuit $D$ that computes the same function with $C$. It is easy to see that (the decision version of) this problem is in $\Sigma_2^{\mathrm{p}} = \mathsf{NP}^{\mathsf{NP}}$. In the case when $\mathfrak{C} = \mathsf{DNF}$, it was shown to be $\Sigma_2^{\mathrm{p}}$-complete by Umans [Uma01], and for a constant depth formula, Buchfuhrer and Umans [BU11] resolved the $\Sigma_2^{\mathrm{p}}$-completeness.

Another variant related to learning theory is the following: Given a sample of points $(x_1, \cdots, x_t)$ and a function value $(f(x_1), \cdots, f(x_t))$ for some unknown partial Boolean function $f$, the task is to find a smallest circuit $C$ that is consistent with $f$. A similar question is studied in the literature of learning theory. For example, Alekhnovich, Braverman, Feldman, Klivans, and Pitassi [ABF+08] showed that learning DNF formulas by using OR-of-thresholds is NP-hard.

### 1.5 Organization and Our Contributions

We outline the organization of this thesis and our contributions below.

**Chapter 2 – Preliminaries**

We start with introducing notation and the problems we consider throughout this thesis. Then we review fundamental properties of Kolmogorov complexity and Kolmogorov randomness. Of particular importance is the interplay between Kolmogorov-randomness and pseudorandomness: every hitting set generator is not secure against any dense set of Kolmogorov-random strings. This property of Kolmogorov complexity plays a crucial role throughout this thesis.

**Chapter 3 – Hardness Based on Average-Case Complexity**

We investigate the average-case complexity of MDLPs. The main technique of this chapter is based on the interplay between Kolmogorov complexity and pseudorandomness. Specifically, we make use of the property of a dense subset of Kolmogorov-random strings, and show that almost all cryptographic primitives can be broken with MINKT oracles. In particular, we show that every auxiliary-input one-way function can be inverted under any MCSP oracle. The result appeared in [AH17].

We then show that MDLPs are hard under popular conjectures about average-case complexity such as the Planted Clique Conjecture and Random 3SAT; these results provide evidence that MDLPs are not likely to be in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Most of these results are presented in [HS17], but the result about Planted Clique is shown here not only for MKTP but also for MCSP.

**Chapter 4 – Non-Black-Box Worst-Case to Average-Case Reductions**

We present the first non-black-box worst-case to average-case reductions within $\mathsf{NP}$. It was shown by Bogdanov and Trevisan [BT06b] that any problem outside $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$ cannot be reduced to an average-case problem within $\mathsf{NP}$ via a (nonadaptive) black-box reduction. Here we show that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ and GapMINKT are reducible to their average-case versions; since these problems are conjectured to be outside of $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$, our results overcome the limits of black-box reductions unless the conjectures fail. The results of this chapter appeared in [Hir18], and received the Machtey Award for Best Student Papers in the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS'18).

**Chapter 5 – Black-box Reductions to Dense Subsets of Random Strings**

While the proofs of Chapter 4 seem to be non-black-box, it might be possible that there could be an alternative proof that yields a black-box reduction, and then, combining limits of black-box reductions, we could conclude that $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{coNP/poly}$, which would refute several conjectures about average-case complexity. Thus it is desirable to clarify in what sense our reductions are non-black-box (instead of just relying on several conjectures). For this purpose, we continue the line of work showing the limits of black-box reductions, and show that black-box reductions to any oracle avoiding any hitting set generator can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$, i.e., without the advice "$/\mathsf{poly}$". We will discuss that the reductions of Chapter 4 are inherently non-black-box *unconditionally* in a certain formal sense. These results are based on an unpublished manuscript with Osamu Watanabe.

**Chapter 6 – Reductions to the Set of Kolmogorov-Random Strings**

We consider a computability-theoretic analogue of MCSP: the set $R_{\mathrm{K}_U}$ of (resource-unbounded) Kolmogorov-random strings. It was conjectured by Allender [All12] that nonadaptive deterministic polynomial-time reductions to $R_{\mathrm{K}_U}$ exactly characterizes $\mathsf{BPP}$ in some sense; his conjecture can be seen as non-$\mathsf{NP}$-hardness of the computability-theoretic analogue of MCSP under such reducibility notions (unless $\mathsf{NP} \subseteq \mathsf{BPP}$). In this chapter we disprove his conjecture under the assumption that the exponential-time hierarchy does not collapse to $\mathsf{BPEXP}$. This chapter is based on an unpublished manuscript of the author.

**Chapter 7 – Hardness of MCSP Implies Circuit Lower Bounds**

Thanks to the non-black-box reductions of Chapter 4, in order to exclude Heuristica, it suffices to prove $\mathsf{NP}$-hardness of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$. Unfortunately, we present several barriers for proving $\mathsf{NP}$-hardness of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$. Here we show that $\mathsf{NP}$-hardness of MCSP under deterministic reductions implies circuit lower bounds; thus it is extremely difficult to prove $\mathsf{NP}$-hardness of MCSP under such reductions. These results first appeared in [HW16], but some of them are improved.

**Chapter 8 – Oracle-Independent Reductions**

While the results of Chapter 7 explain why a *deterministic* reduction is not likely to be used to prove $\mathsf{NP}$-hardness of MCSP, it does not explain why it is difficult to prove $\mathsf{NP}$-hardness of MCSP via a *randomized* reduction. Here we argue this by introducing the new notion of *oracle-independent* reductions to MCSP. We say that a reduction to MCSP is oracle-independent if the reduction can be generalized to $\mathrm{MCSP}^A$ for every oracle $A$. We show that most reduction techniques are oracle-independent, and prove that such reduction techniques cannot reduce any language outside $\mathsf{AM} \cap \mathsf{coAM}$ to MCSP. These results appeared in [HW16].

**Chapter 9 – NP-hardness of MCSP for DNF-XOR Circuits**

Given the barriers for proving $\mathsf{NP}$-hardness for MCSP, we turn our attention to MCSP for *restricted circuit classes*. It was already shown by Masek [Mas79] in 1979 that it is $\mathsf{NP}$-hard to solve MCSP for DNF formulas. Nearly four decades later, no $\mathsf{NP}$-hardness result of MCSP for a circuit class more expressive than $\mathsf{DNF}$ formulas was known. In this chapter, we make the first progress by establishing $\mathsf{NP}$-hardness of MCSP for $\mathsf{DNF} \circ \mathsf{XOR}$ circuits. The result appeared in [HOS18].

**Chapter 10 – Hardness under Local Reductions**

We show that the complexity class $\mathsf{DET}$ is reducible to MKTP via a local reduction such that each output bit of the reduction depends only on a constant number of input bits. In contrast, a previous result of Murray and Williams [MW17] showed that there is no reduction from $\mathsf{PARITY}$ to MCSP or MKTP under $\mathsf{DTIME}(n^{0.49})$ reductions. Our results highlight that the nonuniformity is important for reductions to MKTP. Using similar techniques, we also obtain an equivalence between hardness of $\mathrm{MKTP}^A$ and circuit lower bounds for some class of oracles $A$, thereby presenting a partial converse to Chapter 7. These results appeared in [AH17].

**Chapter 11 – Natural NP-Intermediate Problems**

A problem in $\mathsf{NP}$ is called $\mathsf{NP}$-*intermediate* if it is neither solvable in $\mathsf{P}$ nor $\mathsf{NP}$-complete. It has been known since the work of Ladner [Lad75] that some $\mathsf{NP}$-intermediate problem exists under the weakest assumption that $\mathsf{P} \neq \mathsf{NP}$. However, to the best of our knowledge, no natural $\mathsf{NP}$-intermediate problem that is supported by a reasonable complexity-theoretic argument was known. MCSP is also a prominent candidate for "an $\mathsf{NP}$-intermediate status", in the sense that there is strong evidence against MCSP $\notin \mathsf{P}$, but we do not know any $\mathsf{NP}$-hardness proof for MCSP. Motivated by this, we present the first natural $\mathsf{NP}$-intermediate problems under very weak assumptions such as $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or the existence of auxiliary-input one-way functions. Specifically, approximating the minimum $n$-variable circuit size within a factor of $2^{(1-o(1))n}$ and approximating the maximum clique of an $n$-vertex graph within a factor of $n^{1-o(1)}$ are shown to be $\mathsf{NP}$-intermediate under $\mathsf{P/poly}$-Turing reductions. These results appeared in [AH17].

**Chapter 12 – Unconditional Lower Bounds**

It is widely believed that solving MCSP is hard. Indeed, under cryptographic assumptions, one cannot solve MCSP in polynomial time. However, due to our poor understanding of unconditional circuit lower bounds, it does not necessarily mean that we can prove an unconditional lower bound for MCSP. Here we show several non-trivial circuit lower bounds for MCSP and MKTP. Specifically, we show that MCSP cannot be solved by a de Morgan formula of size $N^{2-o(1)}$, and MKTP cannot be solved by $\mathsf{AC}^0[p]$ circuits on average for any prime $p$. These results appeared in [HS17].

**Chapter 13 – Conclusions**

Finally, we conclude this thesis by highlighting some important open questions and perspective. We will also discuss several subsequent work.

# Chapter 2

# Preliminaries

In this chapter, we review several notations that will be used throughout this thesis, background on complexity theory, and basic properties of Kolmogorov complexity. We refer the reader to standard textbooks such as [AB09, LV08, Vol99, Vad12] for further background.

## 2.1 Basic Notion

For an integer $n \in \mathbb{N}$, let $[n]$ denote $\{1, \cdots, n\}$. For a string $x \in \{0,1\}^n$, we denote by $x_i \in \{0,1\}$ the $i$th bit of $x$ for every $i \in [n]$, and $x_{n+1}$ is defined as $\perp$ (i.e., a special stop symbol). We denote by $|x|$ the length of a string $x$.

Given two strings $x, y \in \{0,1\}^*$, we define the *pairing function* as $\langle x, y \rangle := 1^{|x|}0xy$. This is one of the most common ways to encode two strings into one string. We often write $(x, y)$ instead of $(\langle x, y \rangle)$ when a detail about how to encode a pair is not important. We also abbreviate $\langle x, \langle y, z \rangle \rangle$ as $\langle x, y, z \rangle$. Note that $|\langle x, y \rangle| = 2|x| + |y| + 1$.

We often represent Boolean functions as strings via the truth table mapping. Given a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, the truth table $\mathtt{tt}(f)$ of $f$ is defined as the $2^n$-bit string $f(\underline{1}_n)f(\underline{2}_n)\cdots f(\underline{2^n}_n) \in \{0,1\}^{2^n}$, where $\underline{i}_n$ denotes the $i$th string of $\{0,1\}^n$ in the lexicographic order. Conversely, the inverse function of $\mathtt{tt}$ is denoted by $\mathtt{fn}$. We often identify a Boolean function with its truth table.

For a Turing machine $M$, we denote by $M(x)$ the output of $M$ on input $x \in \{0,1\}^*$.

### 2.1.1 Distribution

For a distribution $\mathcal{D}$, we indicate by $x \sim \mathcal{D}$ that $x$ is a random sample from the distribution $\mathcal{D}$. For a finite set $A$, we indicate by $x \sim A$ that $x$ is a random sample chosen from $A$ uniformly at random.

We denote by $\mathcal{U} := \{\mathcal{U}_n\}_{n\in\mathbb{N}}$ the ensemble of the uniform distributions $\mathcal{U}_n$ on strings of length $n \in \mathbb{N}$.

### 2.1.2 Time Bounds

For a function $t\colon \mathbb{N} \to \mathbb{N}$, we say that $t$ is *efficiently computable* if $t(n)$ is computable in time polynomial in $n$, that is, there exists a polynomial-time machine $M$ such that $M(1^n) = t(n)$ for every $n \in \mathbb{N}$. We say that $t$ is a *polynomial* if $t$ is efficiently computable and there exist constants $\epsilon$ and $c$ such that $0 < \epsilon < c$ and $n^\epsilon \leq t(n) \leq n^c$ for all sufficiently large $n$.

### 2.1.3 Circuit and Its Size

A *circuit* on $n$ variables is a directed acyclic graph such that each internal node is labelled with $\mathsf{AND}$, $\mathsf{OR}$, or $\mathsf{NOT}$, and a node with in-degree 0 is called input gates and labelled with $x_1, x_2, \cdots$, or $x_n$. A circuit $C$ naturally computes a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, and we often identify $C$ with the function $f$ computed by the circuit $C$. There are several ways to measure the "size" of a circuit – the number of gates, wires, or the description length. For each different size measure, we obtain potentially different MCSPs. For concreteness, we define the size of a circuit as the number of gates. For a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, let $\mathsf{size}(f)$ denote the minimum size of a circuit that computes $f$. Abusing notation, for a string $x \in \{0,1\}^{2^n}$, let $\mathsf{size}(x)$ denote the minimum size of a circuit whose truth table is $x$.

### 2.1.4 Languages and Promise Problems

A set $L \subseteq \{0,1\}^*$ of strings is called a *language* or *oracle*. We identify $L$ with its characteristic function $L\colon \{0,1\}^* \to \{0,1\}$ such that $L(x) = 1$ iff $x \in L$ for every $x \in \{0,1\}^*$. For an integer $n \in \mathbb{N}$, let $L^{=n}$ denote $L \cap \{0,1\}^n$.

A *promise problem* is a pair $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ of languages $\Pi_{\text{YES}}, \Pi_{\text{NO}} \subseteq \{0,1\}^*$ such that $\Pi_{\text{YES}} \cap \Pi_{\text{NO}} = \varnothing$, where $\Pi_{\text{YES}}$ and $\Pi_{\text{NO}}$ are regarded as the set of YES and NO instances, respectively. If $\Pi_{\text{YES}} = \{0,1\}^* \setminus \Pi_{\text{NO}}$, we identify $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ with the language $\Pi_{\text{YES}} \subseteq \{0,1\}^*$. We say that a language $A$ *solves* a promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ (or *satisfies* the promise of $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$) if $\Pi_{\text{YES}} \subseteq A \subseteq \{0,1\}^* \setminus \Pi_{\text{NO}}$. For a complexity class $\mathfrak{C}$ such as $\mathsf{ZPP}$ and $\mathsf{BPP}$, we denote by Promise-$\mathfrak{C}$ the promise version of $\mathfrak{C}$.

## 2.2 Complexity Classes

We refer the reader to [AB09, Vol99] for background and more complete definitions of the standard circuit complexity complexity classes such as

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

as well as the standard complexity classes $\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{AM} \subseteq \mathsf{PH} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP} \subseteq \mathsf{NEXP}$. $\mathsf{E}$ denotes $\mathsf{DTIME}(2^{O(n)})$; $\mathsf{EXP}$ denotes $\mathsf{DTIME}(2^{n^{O(1)}})$. The notation $\mathsf{i.o.SIZE}(s(n))$ denotes the class of all languages $A$ such that, for infinitely many lengths $n \in \mathbb{N}$, there is a circuit of size at most $s(n)$ accepting the strings in $A^{=n}$. Similarly, for an oracle $B$, $\mathsf{i.o.SIZE}^B(s(n))$ denotes the class of all languages $A$ such that, for infinitely many lengths $n$, there is a $B$-*oracle circuit* (that is, a circuit that can have oracle gates that answer a query $q \stackrel{?}{\in} B$) of size at most $s(n)$ accepting the strings $A^{=n}$. The *counting hierarchy* $\mathsf{CH}$ [Tor91] consists of the classes $\mathsf{PP}, \mathsf{PP}^{\mathsf{PP}}, \mathsf{PP}^{\mathsf{PP}^{\mathsf{PP}}}$, etc.

### 2.2.1 Reductions

Let $\mathfrak{C}$ be either a class of functions or a class of circuits. We say that $A \leq_{\text{m}}^{\mathfrak{C}} B$ if there is a function $f \in \mathfrak{C}$ (or $f$ computed by a circuit family in $\mathfrak{C}$, respectively) such that $x \in A$ iff $f(x) \in B$. The more powerful notion of Turing reducibility also plays an important role in this thesis. Here, $\mathfrak{C}$ is a complexity class that admits a characterization in terms of Turing machines or circuits, which can be augmented with an "oracle" mechanism, either by providing a "query tape" or "oracle gates". We say that $A \leq_{\text{T}}^{\mathfrak{C}} B$ (or $A \in \mathfrak{C}^B$ if there is an oracle machine in

$\mathfrak{C}$ (or a family of oracle circuits in $\mathfrak{C}$) accepting $A$, when given oracle $B$. Turing reductions that are "nonadaptive" – in the sense that the list of queries that are posed on input $x$ does not depend on the answers provided by the oracle – are called *truth table reductions*, *nonadaptive reductions*, or *reductions with parallel queries*. In this case, we write $A \leq_{\mathrm{tt}}^{\mathfrak{C}} B$ or $A \in \mathfrak{C}_{\parallel}^{B}$. For example, $\mathsf{BPP}_{\parallel}^{R}$ denotes the class of languages solvable by a randomized polynomial-time nonadaptive machine with oracle access to $R$.

## 2.3 Kolmogorov Complexity

We refer the reader to a text by Li and Vitányi [LV08] for background of Kolmogorov complexity. Here we review several variants and important properties of Kolmogorov complexity.

In order to introduce all the variants of Kolmogorov complexity in a unified way, we fix some efficient universal *random access* Turing machine $U$. That is, $U$ efficiently simulates every Turing machine which has random access to its input tape. More specifically, for every random access Turing $M$, there exists a string $d_M \in \{0,1\}^*$ (which encodes a source code of $M$) such that, on input $(d_M, x)$, $U$ outputs $M(x)$ within time $O(t \log t)$ when $M$ outputs $x$ in $t$ steps, for every $x \in \{0,1\}^*$ (cf. [ABK+06b] and the references therein).

The standard (resource-unbounded) Kolmogorov complexity is defined as follows.

**Definition 2.1** (Kolmogorov Complexity)**.** *The Kolmogorov complexity of a string $x \in \{0,1\}^*$ is defined as*

$$\mathrm{K}(x) := \min\{\, |d| \mid \ U(d) = x \ \}.$$

Kolmogorov complexity defines the notion of *randomness* of a finite string in terms of compressibility.

**Definition 2.2** ($r$-random)**.** *Let $r\colon \mathbb{N} \to \mathbb{N}$ be a function, and let $\mathrm{K}_\mu$ be any variant of Kolmogorov complexity. We say that a string $x$ is* $r$-random *with respect to $\mathrm{K}_\mu$ if $\mathrm{K}_\mu(x) \geq r(|x|)$.*

A fundamental property of Kolmogorov-randomness is that almost all strings are random:

**Fact 2.3** (A uniformly random string is Kolmogorov-random)**.** *For any type of Kolmogorov complexity* K*, the number of all the strings $x \in \{0,1\}^*$ such that $\mathrm{K}(x) \leq k$ is at most $2^{k+1}$.*

*Proof.* Any string $x$ with $\mathrm{K}(x) \leq k$ can be *described* by some string $d$ of length $k$ (that is, $U(d) = x$). Thus the set $\{\, x \in \{0,1\}^* \mid \mathrm{K}(x) \leq k \,\}$ is a subset of $\{\, U(d) \mid |d| \leq k \,\}$, and this set is of size at most $\sum_{i=0}^{k} 2^i \leq 2^{k+1}$. □

### 2.3.1 Time-Bounded Kolmogorov Complexity

By restricting the running time of $U$ to a parameter $t$, we obtain the notion of time-bounded Kolmogorov complexity.

**Definition 2.4** (Time-bounded Kolmogorov complexity)**.** *For any string $x \in \{0,1\}^*$ and any integer $t \in \mathbb{N}$, the* Kolmogorov complexity *of $x$ within time $t$ is defined as*

$$\mathrm{K}_t(x) := \min\{\, |d| \mid U(d) = x \text{ in } t \text{ steps } \}.$$

There are several variants of time-bounded Kolmogorov complexity. KT-complexity was proposed in [All01, ABK$^+$06b] in order to model circuit complexity in terms of time-bounded Kolmogorov complexity: it is known that $\mathrm{KT}(\mathtt{tt}(f))$ and the minimum circuit size of $f$ are polynomially-related to each other, by simulating a circuit by a random access machine and vice versa. The KT-complexity of a string $x$ is the minimum of $|d| + t$, where $d$ is a string describing $x$ in time $t$. More formally:

**Definition 2.5** (KT-complexity [All01, ABK$^+$06b])**.** *The* KT*-complexity of $x$ is defined as*

$$\mathrm{KT}(x) := \min\{\, |d| + t \mid \ U(d, i) = x_i \text{ in } t \text{ steps for any } i \in [n+1] \ \}.$$

The fact that the machine $U$ is allowed to have random access to the bits of the description $d$ makes it easier to implement certain algorithms than using the hardware formalisms of circuit complexity. This is why there are some results for MKTP that are not currently known to hold for MCSP.

There is an exponential-time analogue of time-bounded Kolmogorov complexity introduced by Levin [Lev84].

**Definition 2.6** (Levin's Kolmogorov Complexity [Lev84])**.** *The* Levin's Kolmogorov complexity $\mathrm{Kt}(x)$ *of a string $x$ is defined as*

$$\mathrm{Kt}(x) := \min\{\, |d| + \log t \mid U(d) \text{ outputs } x \text{ in time } t \,\}.$$

## 2.4 Minimum Description Length Problems

### 2.4.1 MCSP

We now define MCSP formally.

**Definition 2.7** (Minimum Circuit Size Problem [KC00])**.** *The Minimum Circuit Size Problem, abbreviated as* MCSP*, is defined as follows*:

$$\mathrm{MCSP} := \{\, (\mathtt{tt}(f), s) \in \{0,1\}^* \times \mathbb{N} \mid \mathsf{size}(f) \leq s \,\}.$$

It is also convenient to define a parameterized version of MCSP, i.e., a version whose size parameter is fixed.

**Definition 2.8** (Parameterized Minimum Circuit Size Problem)**.** *Let $s \colon \mathbb{N} \to \mathbb{N}$ be a function. The Minimum Circuit Size Problem with parameter $s$, abbreviated as* MKTP$[s]$*, is defined as follows*:

$$\mathrm{MCSP}[s] := \{\, \mathtt{tt}(f) \in \{0,1\}^* \mid \mathsf{size}(f) \leq s(n) \text{ for } f \colon \{0,1\}^n \to \{0,1\} \ \}.$$

We also define an approximation version of MCSP.

**Definition 2.9** (Approximation version of MCSP)**.** *For any function $\epsilon \colon \mathbb{N} \to [0,1]$, the promise problem* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *is defined as* $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *such that*

$$\Pi_{\mathrm{YES}} := \{\, (\mathtt{tt}(f), s) \mid \mathsf{size}(f) \leq s \,\},$$
$$\Pi_{\mathrm{NO}} := \{\, (\mathtt{tt}(f), s) \mid \mathsf{size}(f) > 2^{(1-\epsilon(2^n))\cdot n} \cdot s \,\},$$

*where $n$ denotes the number of variables of $f$.*

When $\epsilon = 1$, $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is equivalent to MCSP.

There is a natural search version associated to the promise problem.

**Definition 2.10** (Search version of GapMCSP)**.** *For any function* $\epsilon\colon \mathbb{N} \to [0,1]$*, the search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *is defined as follows: Given as input a function* $f\colon \{0,1\}^n \to \{0,1\}$ *represented by its truth table, the task is to output a circuit* $C$ *such that* $C$ *computes* $f$ *and* $|C| \leq 2^{(1-\epsilon(2^n))n} \cdot \mathsf{size}(f)$*.*

**Fact 2.11** (Decision reduces to search for MCSP)**.** *Let* $\epsilon\colon \mathbb{N} \to [0,1]$ *be any efficiently computable function. If there exists a randomized algorithm solving the search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$*, then* $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \text{Promise-}\mathsf{RP}$*.*

*Proof Sketch.* Here is an algorithm for the decision version. On input $(f, s)$, run the search algorithm on input $f$ to obtain some circuit $C$. Accept if and only if $C$ computes $f$ and the circuit size of $C$ is at most $2^{(1-\epsilon(2^n))\cdot n} \cdot s$. □

### 2.4.2 MKTP

A minimum time-bounded Kolmogorov complexity problem closely related to MCSP is called MKTP.

**Definition 2.12** (Minimum Kolmogorov Time-bounded Complexity Problem)**.** *The Minimum Kolmogorov Time-bounded Complexity Problem, abbreviated as* MKTP*, is defined as follows:*

$$\mathrm{MKTP} := \{\, (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathrm{KT}(x) \leq s \,\}.$$

*Let* $s\colon \mathbb{N} \to \mathbb{N}$ *be a function. The parameterized version of* MKTP *is defined as*

$$\mathrm{MKTP}[s] := \{\, x \in \{0,1\}^* \mid \mathrm{KT}(x) \leq s(|x|) \,\}.$$

Similarly, we will consider an exponential-time analogue of MKTP:

**Definition 2.13** (Minimum Kt-Complexity Problem)**.**

$$\mathrm{MKtP} := \{\, (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathrm{Kt}(x) \leq s \,\}.$$

### 2.4.3 MINKT

MINKT is a problem asking for the time-bounded Kolmogorov complexity of $x$ on input $x$ and a time bound $t$.

**Definition 2.14** (Ko [Ko91])**.** *Define*

$$\mathrm{MINKT} := \{\, (x, 1^t, 1^s) \mid \mathrm{K}_t(x) \leq s \,\}.$$

We also define an approximation version of MINKT, parameterized by $\sigma$ and $\tau$.

**Definition 2.15** (An aproximation version of MINKT)**.** *Let* $\sigma, \tau\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *be any functions such that* $\sigma(n,s) \geq s$ *and* $\tau(n,t) \geq t$ *for any* $n, s, t \in \mathbb{N}$*.* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ *is a promise problem* $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *defined as follows.*

$$\Pi_{\mathrm{YES}} := \{\, (x, 1^t, 1^s) \mid \mathrm{K}_t(x) \leq s \,\},$$
$$\Pi_{\mathrm{NO}} := \{\, (x, 1^t, 1^s) \mid \mathrm{K}_{\tau(|x|,t)}(x) > \sigma(|x|, s) \,\}.$$

When $\sigma(n,s) = s$ and $\tau(n,t) = t$, the promise problem $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ coincides with MINKT.

### 2.4.4 Oracle Versions

There are natural oracle versions for each problem and each version of Kolmogorov complexity. For example, for any oracle $A$, the time-bounded Kolmogorov complexity of a string $x$ in time $t$ under oracle $A$ is defined as

$$\mathrm{K}^A_t(x) := \min\{\, |d| \mid U^A(d) = x \text{ in } t \text{ steps } \}.$$

Then we define an oracle version of MINKT as

$$\mathrm{MINKT}^A := \{\, (x, 1^t, 1^s) \mid \mathrm{K}^A_t(x) \leq s \,\}.$$

Similarly, for every oracle $A$, we write oracle versions of MCSP and MKTP by $\mathrm{MCSP}^A$ and $\mathrm{MKTP}^A$, respectively.

## 2.5 Hitting Set Generators

A *hitting set generator* (HSG) is a notion weaker than a pseudorandom generator (PRG). A survey on pseudorandomness can be found in [Vad12].

**Definition 2.16** (dense)**.** *Let $A$ be a set, $\gamma \in [0,1]$, and $\mathcal{D}$ be a distribution. We say that $A$ is* $\gamma$-dense *over $\mathcal{D}$ if $\Pr_{w\sim\mathcal{D}}[w \in A] \geq \gamma$. We abbreviate $\mathcal{D}$ when $\mathcal{D}$ is the uniform distribution on $\{0,1\}^n$ and $A \subseteq \{0,1\}^n$.*

A hitting set generator is defined as follows.

**Definition 2.17** (Hitting set generators)**.** *Let $\gamma\colon \mathbb{N} \to [0,1]$ be a function. Let $G = \{G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^{t(n)}\}_{n\in\mathbb{N}}$ be a family of functions. A promise problem $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ is said to* $\gamma$-avoid *$G$ over an ensemble of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ if for all sufficiently large $n \in \mathbb{N}$,*

- *$G_n(z) \in \Pi_{\mathrm{NO}}$ for any $z \in \{0,1\}^{s(n)}$, and*
- *$\Pr_{w\sim\mathcal{D}_n}\big[w \in \Pi_{\mathrm{YES}}\big] \geq \gamma(n)$ (i.e., $\Pi_{\mathrm{YES}}$ is $\gamma(n)$-dense over $\mathcal{D}_n$).*

*By default, we consider the uniform distribution over $\{0,1\}^{t(n)}$ as the distribution $\mathcal{D}_n$. A function family $G$ is called a* hitting set generator $\gamma$-secure against a complexity class $\mathfrak{C}$ *if there is no promise problem $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}}) \in \mathfrak{C}$ that $\gamma$-avoids $G$.*

For a hitting set generator, we measure the time complexity with respect to the output length $t(n)$; that is, we say that a family of functions $G := \{G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^{t(n)}\}_{n\in\mathbb{N}}$ is *efficiently computable* if there exists a polynomial-time algorithm that, on input $z \in \{0,1\}^{s(n)}$, computes $G_n(z)$ in time $\mathsf{poly}(t(n))$ for all large $n \in \mathbb{N}$.

Natural properties, introduced by Razborov and Rudich [RR97], can be cast as algorithms avoiding a particular hitting set generator. The hitting set generator is defined as follows.

**Definition 2.18** (Circuit interpreter)**.** *Let $s\colon \mathbb{N} \to \mathbb{N}$ be a function. Let*

$$G^{\mathsf{int},s} := \{G^{\mathsf{int},s}_\ell\colon \{0,1\}^{O(s(\ell)\log s(\ell))} \to \{0,1\}^{2^\ell}\}_{\ell\in\mathbb{N}}$$

*denote the family of* circuit interpreters $G^{\mathsf{int},s}_\ell$ *with parameter $s$, defined as follows: $G^{\mathsf{int},s}_\ell$ takes as input a description $z \in \{0,1\}^{O(s(\ell)\log s(\ell))}$ of a circuit $C_z$ of size at most $s(\ell)$ on $\ell$ inputs, and outputs the truth table of the function computed by $C_z$.*

**Definition 2.19** (Γ-natural property)**.** *A promise problem* $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ *is called a* natural property useful against $\mathsf{SIZE}(s(\ell))$ with largeness $\gamma$ *if* $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ *$\gamma$-avoids the circuit interpreter $G^{\mathsf{int},s}$ with parameter $s$. If, in addition,* $(\Pi_{\text{YES}}, \Pi_{\text{NO}}) \in \text{Promise-}\Gamma$ *for a complexity class $\Gamma$ such as $\mathsf{P}$, $\mathsf{BPP}$ or $\mathsf{NP}$, then* $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ *is called a* Γ-natural property*.*

It is easy to see that there exists a $\mathsf{P}^{\text{MCSP}}$-natural property. In fact, it can be shown that a natural property useful against $\mathsf{SIZE}(s(n))$ is essentially equivalent to an errorless heuristic algorithm for MCSP$[s]$ (cf. Lemma 4.36).

A natural property can be defined for restricted circuit classes. Razborov and Rudich [RR97] observed that from almost all circuit lower bound proofs against a circuit class $\mathfrak{C}$, one can *naturally* extract a $\mathsf{P}$-natural property useful against $\mathfrak{C}$. For example, by inspecting circuit lower bound proofs for showing $\mathsf{PARITY} \notin \mathsf{AC}^0$ [Hås86] and $\mathsf{MAJORITY} \notin \mathsf{AC}^0[p]$ [Raz87, Smo87], Razborov and Rudich obtained $\mathsf{P}$-natural properties useful against $\mathsf{AC}^0$ and $\mathsf{AC}^0[p]$. On the other hand, there exists a "pseudorandom function generator" computable in $\mathsf{TC}^0$ conjectured to be secure [NR04] (in our terminology, there exists a hitting set generator $G^{\mathsf{int}}$ secure against $\mathsf{P}/\mathsf{poly}$, where $G^{\mathsf{int}}$ takes a description of $\mathsf{TC}^0$ circuits and outputs the truth table of the function computed by the $\mathsf{TC}^0$ circuit); thus there is no natural property useful against $\mathsf{TC}^0$ assuming the security of the pseudorandom function generator. This explains the natural proof barrier and why no strong circuit lower bound for $\mathsf{TC}^0$ is known.

## 2.6 Kolmogorov Complexity and Pseudorandomness

There is a fundamental relationship between Kolmogorov complexity and hitting set generators: *Pseudorandomly generated strings are not Kolmogorov-random.* Indeed, take any computable generator $G\colon \{0,1\}^s \to \{0,1\}^n$ with $s \ll n$ that purports to extend a seed $z \in \{0,1\}^s$ to a longer sequence $G(z)$ that looks random. Then, for every $z \in \{0,1\}^s$, the output $G(z)$ of the generator $G$ can be described by using the seed $z$ and the program that computes $G$, and hence $\mathrm{K}(G(z)) \leq s + O(1)$; therefore, $G(z)$ is not Kolmogorov-random.

A similar relationship holds for time-bounded Kolmogorov complexity. We claim that any efficiently computable hitting set generator is not secure against a polynomial-time algorithm with one-query oracle access to MINKT: For any efficiently computable hitting set generator $G$ and any seed $s$, $G(s)$ can be described by $s$ and an efficient algorithm in time $t$, and thus $\mathrm{K}_t(G(s))$ is small. On the other hand, for a random $r \sim \{0,1\}^{t(n)}$, the simple counting argument of Fact 2.3 shows that $\mathrm{K}_t(r)$ is large with high probability. Thus the set of all the Kolmogorov-random strings avoids the hitting set generator $G$. (In fact, the same argument works for any dense subset of Kolmogorov-random strings.)

We formally state this fact in the next theorem. It will be convenient to observe that any hitting set generator associated with not only the uniform distribution but also any distribution with sufficiently large min-entropy can be avoided by the set of Kolmogorov-random strings. Here recall that for a distribution $\mathcal{D}$, the *min-entropy* of $\mathcal{D}$ is defined as $\min\{ -\log \Pr_{x\sim\mathcal{D}}[x = x_0] \mid x_0 \in \mathsf{supp}(D) \}$.

**Theorem 2.20** (Pseudorandomness is not Kolmogorov-random)**.** *Let* $G = \{G_n\colon \{0,1\}^{s(n)} \to \{0,1\}^n\}_{n\in\mathbb{N}}$ *be any family of functions computable in time $t(n)$, where $s\colon \mathbb{N} \to \mathbb{N}$ is an efficiently computable function. Let $\gamma\colon \mathbb{N} \to [0,1)$ and $k\colon \mathbb{N} \to [0,\infty)$ be arbitrary functions. Let* $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ *be the promise*

*problem defined as*

$$\Pi_{\text{No}} := \{\, x \in \{0,1\}^* \mid \mathrm{K}_{t(|x|)}(x) \leq s(|x|) + 3\log|x| \,\},$$
$$\Pi_{\text{Yes}} := \{\, x \in \{0,1\}^* \mid \mathrm{K}(x) > k(|x|) - 1 - \log(1/(1-\gamma(|x|))) \,\}.$$

*Then $\Pi$ $\gamma$-avoids $G$ over any ensemble of distributions $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$, where $\mathcal{D}_n$ is a distribution on $\{0,1\}^n$ with min-entropy $k(n)$.*

*Proof.* Since $G_n(z)$ can be described by its seed $z \in \{0,1\}^{s(n)}$ and an integer $n \in \mathbb{N}$ in time $\mathsf{poly}(n)$, we have $\mathrm{K}_t(G_n(z)) \leq s(n) + 2\log n + O(1)$. (Here we used the fact that a pair of $n$ and $z$ can be encoded as a binary string of length $2\log n + s(n) + O(1)$.) Therefore, for a sufficiently large $n \in \mathbb{N}$, it holds that $\mathrm{K}_t(G_n(z)) \leq s(n) + 2\log n + O(1) \leq s(n) + 3\log n$, and thus $G_n(z) \in \Pi_{\text{No}}$.

It remains to claim that $\Pi_{\text{Yes}}$ is $\gamma(n)$-dense with respect to $\mathcal{D}_n$, that is, $\Pr_{w\sim\mathcal{D}_n}[w \in \Pi_{\text{Yes}}] \geq \gamma(n)$. Indeed, for every $n \in \mathbb{N}$,

$$\begin{aligned}
&\Pr_{w\sim\mathcal{D}_n}[w \notin \Pi_{\text{Yes}}] \\
&\leq \sum_{w_0 \in \{0,1\}^n \setminus \Pi_{\text{Yes}}} \Pr_{w\sim\mathcal{D}_n}[w = w_0] \\
&\leq 2^{k(n)+\log(1-\gamma(n))} \cdot 2^{-k(n)} \;=\; 1-\gamma(n),
\end{aligned}$$

where, in the last inequality, we used the fact that the min-entropy of $\mathcal{D}_n$ is at least $k(n)$, and that $|\{\, w_0 \in \{0,1\}^* \mid \mathrm{K}(w_0) \leq k' \,\}| \leq 2^{k'+1}$ (by Fact 2.3). □

# Chapter 3

# Hardness Based on Average-Case Complexity

In this chapter, we show hardness of MDLPs based on average-case complexity. The main technique is based on the fact that MDLPs can be used as a distinguisher for an efficiently computable pseudorandom generator. In Section 3.1, we show that every auxiliary-input one-way function can be inverted by a randomized polynomial-time algorithm with oracle access to MCSP. In particular, MCSP is intractable unless every auxiliary-input one-way function can be inverted. In Section 3.2, we present several hardness results based on popular conjectures about average-case complexity.

## 3.1 Auxiliary-Input One-Way Functions and MCSP

In this section, we investigate the relationship between cryptographic primitives and MCSP. In particular, we show that every auxiliary-input one-way function can be inverted with MCSP oracles. The proof is based on a sequence of constructions of cryptographic primitives and security proofs. Here is an overview: Håstad, Impagliazzo, Levin and Luby [HILL99] constructed a pseudorandom generator from any one-way function. Goldreich, Goldwasser and Micali [GGM86] showed that any pseudorandom generator can be used to construct a pseudorandom *function* generator. Razborov and Rudich [RR97] showed that a natural property can be used to break any pseudorandom function generator. Combining these results, any auxiliary-input one-way function can be inverted with oracle access to any natural property (and in particular MCSP).

Roughly speaking, a one-way function (OWF) $f$ is a polynomial-time computable function such that $f$ is hard to invert on average: no efficient algorithm can find $x' \in f^{-1}(f(x))$ with high probability over a random input $x$. An auxiliary-input one-way function (AIOWF) $f$ is a family of functions $f = \{f_x\}_{x\in\{0,1\}^*}$ such that, for every efficient algorithm $A$, there exists an infinite index set $I_A \subseteq \{0,1\}^*$ such that $A$ fails to invert $f_x$ on average for every $x \in I$.

**Definition 3.1** (Auxiliary-input function)**.** *An* auxiliary-input function *is a family of functions $f = \{f_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}_{x\in\{0,1\}^*}$, where $p$ and $q$ are polynomials. We say that $f$ is* polynomial-time computable *if there exists a polynomial-time algorithm $A$ such that $A(x,y) = f_x(y)$ for every $x \in \{0,1\}^*$ and $y \in \{0,1\}^{p(|x|)}$.*

**Definition 3.2** (Auxiliary-input one-way function (AIOWF))**.** *We say that a randomized algorithm $A$* inverts *an auxiliary-input function $f = \{f_x :$*

$\{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}$ *with success probability* $\delta\colon \mathbb{N} \to [0,1]$ *if*

$$\Pr_{A,\, y \sim \{0,1\}^{p(|x|)}}[A(x, f_x(y)) \in f_x^{-1}(f_x(y))] \geq \delta(|x|)$$

*for every* $x \in \{0,1\}^*$*, where the probability is taken for* $y$ *and the internal coin flips of* $A$*.*

*A randomized algorithm* $A$ *is said to* weakly invert $f$ *if* $A$ *inverts* $f$ *with success probability* $1/r(n)$ *for some polynomial* $r(n)$*. Similarly,* $A$ *is said to* strongly invert $f$ *if, for every parameter* $t > 0$*,* $A(\text{-}, 1^t)$ *inverts* $f$ *with success probability* $1 - 1/t$*. We say that an auxiliary-input function* $f$ *is weakly (resp. strongly) one-way if there is no randomized polynomial-time algorithm* $A$ *strongly (resp. weakly) inverts* $f$*.*

It is possible to construct some strongly one-way function from any weakly one-way function $f$ by amplifying the hardness of $f$. We define $f'$ as the direct product of $f$, that is, $f'(y_1, \cdots, y_t) := (f(y_1), \cdots, f(y_t))$ for some appropriately chosen parameter $t$. Intuitively, in order to invert $f'$, one needs to invert $t$ copies of $f$ simultaneously, and thus it is much harder to invert $f'$ than $f$. Indeed, it can be shown that given any oracle weakly inverting $f'$, one can strongly invert $f$ efficiently:

**Lemma 3.3** (∃ strong OWF ⇔ ∃ weak OWF; Yao [Yao82])**.** *Let* $f = \{f_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{q(|x|)}\}$ *be any polynomial-time computable auxiliary-input function. Define an auxiliary-input function* $f'$ *as*

$$f'_{x,1^t}(y_1, \cdots, y_t) = (f(y_1), \cdots, f(y_t))$$

*for every* $(y_1, \cdots, y_t) \in (\{0,1\}^{p(|x|)})^t$ *and* $t \in \mathbb{N}$*. There exists a randomized polynomial-time oracle machine* $M$ *such that* $M^A$ *strongly inverts* $f$ *for any oracle* $A$ *that weakly inverts* $f'$*.*

A pseudorandom generator is a function $G$ such that $G(z)$ for a random seed $z$ and the uniform distribution are indistinguishable by any efficient algorithm:

**Definition 3.4** (Distinguisher and pseudorandom generator (PRG))**.** *Let* $G = \{G_n : \{0,1\}^{s(n)} \to \{0,1\}^n\}$ *be a family of functions. For a function* $\delta\colon \mathbb{N} \to [0,1]$*, a randomized algorithm* $A$ *is called a* $\delta$-distinguisher *for* $G$ *if* $s(n) < n$ *and*

$$|\Pr_{A,\, z \sim \{0,1\}^{s(n)}}[A(G(z)) = 1] - \Pr_{A,\, w \sim \{0,1\}^n}[A(w) = 1]| \geq \delta(n)$$

*for all but finitely many* $n \in \mathbb{N}$*. We often abbreviate* $\delta$ *and simply say that* $A$ distinguishes $G$ *(from the uniform distribution) if* $A$ *is a* $1/r(n)$*-distinguisher for* $G$ *for some polynomial* $r(n)$*.*

$G$ *is called a* pseudorandom generator *if there exists no randomized polynomial-time algorithm that distinguishes* $G$*.*

We mention that in the context of cryptography, it is more common to define the security of cryptographic primitives with respect to an algorithm that works *infinitely often*; that is, $G$ is called a pseudorandom generator secure against infinitely often adversaries if there is no randomized polynomial-time algorithm that distinguishing $G_n$ for infinitely often $n$. However, for our purpose, it is more natural to consider a pseudorandom generator secure against almost everywhere adversaries as in Definition 3.4.

Håstad, Impagliazzo, Levin and Luby [HILL99] constructed a pseudorandom generator $G^{\mathrm{HILL}(f)}$ from any one-way function $f$, thereby establishing the equivalence between the existence of a pseudorandom generator and that of a one-way function. The proof goes by showing a reduction from weakly inverting a one-way function to distinguishing a pseudorandom generator. Moreover, the reduction carries over to the setting of auxiliary-input one-way functions.

**Lemma 3.5** ($\exists$ OWF $\Leftrightarrow$ $\exists$ PRG; Håstad, Impagliazzo, Levin and Luby [HILL99])**.** *Let $f$ be any polynomial-time computable auxiliary-input function. Then, there exists a polynomial-time computable auxiliary-input function $G^{\mathrm{HILL}(f)} = \{G_x^{\mathrm{HILL}(f)} : \{0,1\}^{p(|x|)} \to \{0,1\}^{2p(|x|)}\}$ satisfying the following: For any polynomial $r(n)$, there exists a randomized polynomial-time oracle machine $M$ such that $M^A$ weakly inverts $f$ for any oracle $A$ such that $A(x,\text{-})$ $1/r(|x|)$-distinguishes $G_x^f$ for every $x \in \{0,1\}^*$.*

Goldreich, Goldwasser and Micali [GGM86] showed how to construct a *pseudorandom function generator* (PRFG) from any pseudorandom generator. A pseudorandom function generator $F$ is a function that takes a seed $z$ and returns a function $F_z$ such that any randomized polynomial-time oracle algorithm with oracle access to $F_z$ cannot distinguish $F_z$ from a truly random function when the seed $z$ is chosen uniformly at random. The truth table of $F$ gives us a pseudorandom generator $G^{\mathrm{GGM}}$ such that each bit of $G^{\mathrm{GGM}}(z)$ is efficiently computable. Specifically, for any pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$, let $G_0(z)$ be the first $n$ bits of $G(z)$, and let $G_1(z)$ be the second $n$ bits of $G(z)$, so that $G(z) = G_0(z)G_1(z)$. Let $k \in \mathbb{N}$ be an arbitrary parameter. Define a function $F_z$ so that $F_z(w)$ is the first bit of $G_{w_1}(G_{w_2}(\cdots(G_{w_k}(z))\cdots))$ for every $w \in \{0,1\}^k$. Then we define $G^{\mathrm{GGM}}(z) \in \{0,1\}^{2^k}$ as the truth table of the function $F_z$. By a standard hybrid argument (cf. [RR97]), it can be shown that $G^{\mathrm{GGM}}$ is a pseudorandom generator if $G$ is a pseudorandom generator. Moreover, since each bit of $G^{\mathrm{GGM}}(z)$ is polynomial-time computable given $z$ and an index, there exists a universal constant $c$ such that $\mathsf{size}(G^{\mathrm{GGM}}(z)) \leq (n+k)^c + c$. (Here note that a string $G^{\mathrm{GGM}}(z)$ is interpreted as a truth table of a function.) In particular, for any constant $\epsilon > 0$, we can take a large enough $k = O(\log n)$ so that $\mathsf{size}(G^{\mathrm{GGM}}(z)) \leq 2^{\epsilon k}$.

**Lemma 3.6** ($\exists$ PRFG $\Leftrightarrow$ $\exists$ PRG; Goldreich, Goldwasser and Micali [GGM86]; Razborov and Rudich [RR97])**.** *For any constant $\epsilon > 0$, there exists an efficiently computable function $k\colon \mathbb{N} \to \mathbb{N}$ with $k(n) = O(\log n)$ satisfying the following: For any polynomial-time computable auxiliary-input function $G = \{G_x : \{0,1\}^{p(|x|)} \to \{0,1\}^{2p(|x|)}\}$, there exist a polynomial-time computable auxiliary-input function $G^{\mathrm{GGM}} = \{G_x^{\mathrm{GGM}} : \{0,1\}^{p(|x|)} \to \{0,1\}^{2^{k(|x|)}}\}$ and a randomized polynomial-time oracle machine $M$ such that $M^A(x,\text{-})$ distinguishes $G_x$ for every $x$ and every oracle $A$ that distinguishes $G_x^{\mathrm{GGM}}$. Moreover, $\mathsf{size}(G_x^{\mathrm{GGM}}(z)) \leq 2^{\epsilon \cdot k(|x|)}$ holds for every $x \in \{0,1\}^*$ and every $z \in \{0,1\}^{p(|x|)}$.*

Combining these ingredients mentioned above, we obtain the following:

**Theorem 3.7** (Inverting any AIOWF under a natural property oracle)**.** *Let $f$ be any polynomial-time computable auxiliary-input function. Let $c \in \mathbb{N}$ and $\epsilon \in [0,1)$ be arbitrary constants. There exists a randomized polynomial-time oracle machine $M$ satisfying the following: For any natural property $R \subseteq \{0,1\}^*$ useful against $\mathsf{SIZE}(2^{\epsilon n})$ with largeness $2^{-cn}$, $M^R$ strongly inverts $f$.*

*Proof.* We construct a candidate auxiliary-input pseudorandom generator $G$ as follows: First, as in Lemma 3.3, we define $f'$ as the direct product of $f$. Second, by using Lemma 3.5, we construct a candidate pseudorandom generator $G^{\mathrm{HILL}(f')}$ from the candidate one-way function $f'$. Finally, by using Lemma 3.6, we construct a candidate pseudorandom function generator $G := (G^{\mathrm{HILL}(f')})^{\mathrm{GGM}}$ from $G^{\mathrm{HILL}(f')}$.

We now argue that the natural property $R$ $2^{-ck(n)}$-distinguishes $G$: On one hand, by Lemma 3.6, we have $\mathsf{size}(G_x(z)) \leq 2^{\epsilon k(|x|)}$ for every $x \in \{0,1\}^*$ and $z \in \{0,1\}^{p(|x|)}$; thus $G_x(z) \notin R$. On the other hand, by the largeness of $R$, we have $\Pr_{w \sim \{0,1\}^{2^{k(|x|)}}}[w \in R] \geq 2^{-c \cdot k(|x|)}$. Therefore, we obtain

$$\left| \Pr_{w \sim \{0,1\}^{2^{k(|x|)}}}[w \in R] \quad - \Pr_{z \sim \{0,1\}^{p(|x|)}}[G(z) \in R] \right| \geq 2^{-c \cdot k(|x|)}.$$

By Lemma 3.6, we obtain a randomized polynomial-time algorithm $M_1$ such that $M_1^R$ distinguishes $G^{\mathrm{HILL}(f')}$. Using $M_1^R$ as an oracle $A$ in Lemma 3.5, we obtain a randomized polynomial-time algorithm $M_2$ such that $M_2^R$ weakly inverts $f'$. Similarly, using $M_2^R$ as an oracle $A$ in Lemma 3.3, we obtain a randomized polynomial-time algorithm $M_3$ such that $M_3^R$ strongly inverts $f$. □

Since any MCSP oracle can be used as a natural property, we obtain the following immediately:

**Corollary 3.8.** *If* MCSP $\in \mathsf{BPP}$ *then there exists no auxiliary-input one-way function.*

Ostrovsky and Wigderson [OW93] showed that $\mathsf{CZK} \not\subseteq \mathsf{BPP}$ implies the existence of auxiliary-input one-way function, where $\mathsf{CZK}$ ($\supseteq \mathsf{SZK}$) denotes the class of problems with a computational zero knowledge proof system. Similarly, Ostrovsky [Ost91] implicitly showed that $\mathsf{SZK} \not\subseteq \mathsf{BPP}$ implies the existence of auxiliary-input one-way function. Thus by Corollary 3.8, MCSP is harder than $\mathsf{SZK}$ in the sense that MCSP $\in \mathsf{BPP}$ implies $\mathsf{SZK} \subseteq \mathsf{BPP}$. More precisely, in the context of MCSP, it was first shown by Allender and Das [AD17] that $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathrm{MCSP}}$. Based on Theorem 3.7, their result follows from the earlier work of Ostrovsky.

**Lemma 3.9** (implicit in Ostrovsky [Ost91]; explicitly stated in Vadhan [Vad06])**.** *For every problem* $\Pi$ *in* $\mathsf{SZK}$*, there exist a polynomial-time computable auxiliary-input function* $f$ *and a randomized polynomial-time oracle machine* $M$ *such that, for any oracle* $A$ *strongly inverting* $f$*,* $M^A$ *solves* $\Pi$ *with high probability.*

By Lemma 3.9 and Theorem 3.7, we immediately obtain an alternative proof of $\mathsf{SZK}$-hardness of MCSP.

**Corollary 3.10** (Allender and Das [AD17])**.** $\mathsf{SZK} \subseteq \mathsf{BPP}^{\mathrm{MCSP}}$.

This is the best worst-case hardness result for MCSP. We mention that $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ [For89, AH91], and under the standard derandomization hypothesis [KvM02], we have $\mathsf{AM} = \mathsf{NP}$ and thus $\mathsf{SZK} \subseteq \mathsf{NP} \cap \mathsf{coNP}$; therefore, the $\mathsf{SZK}$-hardness cannot be regarded as evidence that MCSP $\notin \mathsf{coNP}$.

**Open Question 3.11.** Provide evidence that MCSP $\notin \mathsf{coNP}$ under any worst-case complexity assumptions.

We mention that there is a fundamental reason why any approach based on Theorem 3.7 is not likely to resolve Open Question 3.11: Akavia, Goldreich, Goldwasser, and Moshkovitz [AGGM06, AGGM10] showed that a randomized polynomial-time nonadaptive black-box reduction from any worst-case problem $L$ to inverting any one-way function can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$, and thus $L \in \mathsf{AM} \cap \mathsf{coAM}$. In the next section, we will sidestep this issue by starting from *average-case* complexity assumptions, and thereby we will present evidence that MCSP $\notin \mathsf{coNP}$.

## 3.2 Average-case Complexity and MDLPs

In this section, we establish hardness of MDLPs based on popular hypotheses on average-case complexity of various problems. Previously, Rudich [Rud97] conjectured that there is no $\mathsf{NP/poly}$-natural property useful against $\mathsf{P/poly}$ (and, in particular, that MCSP $\notin \mathsf{coNP/poly}$). His conjecture is based on a pseudorandom generator constructed by using some average-case hardness of the subset sum problem [IN96]. Here we investigate hardness of MDLPs based on other popular conjectures on average-case complexity.

### 3.2.1 Planted Clique Hardness of MCSP

The *planted clique problem* is one of well studied average-case complexity problems. The problem was suggested independently by Jerrum [Jer92] and Kučera [Kuc95]. Let $n$ be the number of vertices, and $k$ be the size of a planted clique. The input is a random graph with a planted clique of size $k$: that is, we pick an $n$-vertex Erdős-Rényi random graph $G$ (i.e., a random graph where every edge is added with probability $\frac{1}{2}$ independently), pick the set $S$ of $k$ vertices from the $n$ vertices uniformly at random, and add the clique supported on $S$ to the graph $G$. The task is to find a clique of size $k$ in polynomial time.

A search version of a *planted clique conjecture* asserts that there is no randomized polynomial-time algorithm that solves the planted clique problem for any parameter $k$ with $2 \log n \ll k \ll \sqrt{n}$ with high probability (over the choice of a planted random graph and internal coin flips of the randomized algorithm). Indeed, despite intense effort (e.g. [Kuc95, AKS98, FK00]), the state-of-the-art polynomial-time algorithm of Alon, Krivelevich and Sudakov [AKS98] works only for $k = \epsilon\sqrt{n}$ for any constant $\epsilon > 0$. Moreover, it is known that several types of polynomial-time algorithms cannot solve the planted clique problem for $k = n^{1/2-\delta}$ for any constant $\delta > 0$ (cf. [Jer92, BHK$^+$16]).

We now present a reduction from the planted clique problem to MCSP.

**Theorem 3.12** (Planted Clique Hardness of MCSP)**.** *There is a randomized polynomial-time oracle machine that, given a parameter $1^t$, solves the planted clique problem with oracle access to* MCSP *with probability at least $1 - 1/t$.*

*Proof.* Juels and Peinado [JP00] proposed a simple one-way function based on a variant of the planted clique conjecture: A candidate one-way function $f_n$ is defined as $f_n(G, S) :=$ (the union of $G$ and the clique supported on $S$) for a graph $G$ and a set $S$ of $k$ vertices (For simplicity, we abbreviate it as $G \cup S$ below). By Theorem 3.7, there exists a randomized polynomial-time machine $M$ with an MCSP oracle that can strongly invert $f$. Note that the distribution $f_n(G, S)$ where $G$ and $S$ are chosen uniformly at random corresponds to the input distribution of the planted clique problem. Therefore, given a random planted clique $f_n(G, S)$, the algorithm $M^{\mathrm{MCSP}}(f_n(G, S), 1^t)$ outputs $(G', S')$ such that

$G \cup S = G' \cup S'$ with probability $1 - 1/t$. This means that $S'$ is a clique of size $k$ in the graph $G$.

We note that the argument above ignores the detail that $(G, S)$ may not be exactly encoded as a binary string. (And thus it is not clear how to define a Boolean function $f_n$.) This can be easily fixed as follows: While it may not be possible to sample the input distribution of the planted clique problem *exactly*, for any given parameter $1^t$, one can sample a distribution $2^{-t}$-close to the input distribution by using random $\mathsf{poly}(t, n)$ bits in polynomial time. (Here the closeness is in the sense of the statistical distance: we say that two distributions $\mathcal{D}_1, \mathcal{D}_2$ are $\epsilon$-close if, for every function $A\colon \{0,1\}^* \to \{0,1\}$, it holds that $|\mathbb{E}[A(\mathcal{D}_1) - A(\mathcal{D}_2)]| \leq \epsilon$.) Thus the same argument works with an additional failure probability of $2^{-t}$. □

To the best of our knowledge, there is no coNP algorithm that solves the planted clique problem. Thus Theorem 3.12 can be also seen as some evidence that MCSP $\notin$ coNP.

### 3.2.2 Random 3SAT Hardness of MKTP

*Random 3SAT* is another widely investigated problem in the literature of average-case complexity. The problem is defined as follows:

**Definition 3.13** (Random 3SAT)**.** *Let $m\colon \mathbb{N} \to \mathbb{N}$ be a function. Let $\mathcal{D}^{\mathrm{R3SAT}} = \{\mathcal{D}_n^{\mathrm{R3SAT}}\}_{n\in\mathbb{N}}$ be an ensemble of the following distributions $\mathcal{D}_n^{\mathrm{R3SAT}}$ of 3CNF formulas: Let $n$ be the number of variables, and $m = m(n)$ be the number of clauses. The distribution $\mathcal{D}_n^{\mathrm{R3SAT}}$ samples a random $n$-variable $m$-clause 3CNF formula $\varphi$ by choosing each clause independently and uniformly at random from all the possible $2^3\binom{n}{3}$ width-3 clauses on $n$ variables. Given a 3CNF formula, the task of* Random 3SAT *is to accept every satisfiable formula, and reject most formulas sampled from $\mathcal{D}_n^{\mathrm{R3SAT}}$.*

Note that a simple probabilistic argument shows that, for $m = \Delta n$ where $\Delta$ is a sufficiently large constant, most formulas are unsatisfiable. Thus it is trivial to solve Random 3SAT by an algorithm that is allowed to err by simply rejecting every formula. The definition above does not allow such an algorithm, by requiring that the algorithm never makes any error on satisfiable formulas.

Feige and Ofek [FO07] presented a deterministic polynomial-time algorithm that solves Random 3SAT for $m > O(n^{3/2})$. In the case of a nondeterministic polynomial-time algorithm, a much better algorithm is known: Feige, Kim and Ofek [FKO06] showed a coNP algorithm that solves Random 3SAT for $m > O(n^{7/5})$. However, it is still far from the case of a linear number of clauses (i.e., $m = \Theta(n)$).

Ryan O'Donnell (personal communication; cf. [BGSV16]) conjectured that there is no coNP algorithm solving Random 3SAT with $m = \Delta n$ clauses for a sufficiently large constant $\Delta$. We will show that, under this conjecture, MINKT $\notin$ coNP and moreover MKTP $\notin$ coNP.

#### Random 3SAT and a Hitting Set Generator

We observe that any algorithm solving Random 3SAT can be regarded as avoiding some hitting set generator. Indeed, we can define a hitting set generator $G^{\mathrm{R3SAT}}$ so that the image of $G^{\mathrm{R3SAT}}$ contains all the satisfiable formulas.

**Proposition 3.14.** *There exists an efficiently computable family of functions*

$$G^{\mathrm{R3SAT}} = \{G_n^{\mathrm{R3SAT}} : \{0,1\}^{n+\lceil m(n)\log(7\binom{n}{3})\rceil} \to \{0,1\}^{\lceil m(n)\log(8\binom{n}{3})\rceil}\}_{n\in\mathbb{N}}$$

*such that the image of $G_n^{\mathrm{R3SAT}}$ contains all the satisfiable $n$-variable $m(n)$-clause 3CNF formula. (Here we regard the output of $G_n^{\mathrm{R3SAT}}$ as an encoding of a 3CNF formula.)*

*Proof.* Fix any $n \in \mathbb{N}$, and fix any satisfiable $n$-variable $m(n)$-clause 3CNF formula $\varphi$. Let $a \in \{0,1\}^n$ be some satisfying assignment of $\varphi$. We claim that $\varphi$ can be described by the assignment $a$ and some auxiliary information $d \in \{0,1\}^{\lceil m(n)\log(7\binom{n}{3})\rceil}$. ($G_n^{\mathrm{R3SAT}}(a,d)$ is defined as the output of a description procedure below.)

Indeed, given a satisfying assignment $a$ of $\varphi$, there are $(2^3-1)\binom{n}{3}$ possible clauses that are satisfied by $a$. More specifically, fix any 3 variables of a clause; then there are 7 ways to negate these variables so that the resulting clause is satisfied by $a$. (For example, if the assignment $a$ is $\{x_1 \mapsto 0, x_2 \mapsto 1, x_3 \mapsto 0\}$, then $x_1 \vee \neg x_2 \vee x_3$ is the unique clause that is not satisfied by $a$.) Therefore, each clause of $\varphi$ can be described with $\log(7\binom{n}{3})$ bits, and hence $\varphi$ can be described with some auxiliary information $d$ of length $m(n)\log(7\binom{n}{3})$. □

**Theorem 3.15** (Random 3SAT and $G^{\mathrm{R3SAT}}$)**.** *Let $\Pi = (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ be any promise problem that $\gamma$-avoids $G^{\mathrm{R3SAT}}$ over the distribution $\mathcal{D}^{\mathrm{R3SAT}}$. Then the complement $(\Pi_{\mathrm{NO}}, \Pi_{\mathrm{YES}})$ of $\Pi$ solves Random 3SAT: that is, every satisfiable formula is in $\Pi_{\mathrm{NO}}$, and a $\gamma(n)$-fraction of all formulas is in $\Pi_{\mathrm{YES}}$.*

*Proof.* Fix any $n \in \mathbb{N}$. By Proposition 3.14, for any satisfiable $n$-variable formula $\varphi$, there exists a description $(d,a)$ such that $G_n^{\mathrm{R3SAT}}(d,a) = \varphi$; since $\Pi$ avoids $G^{\mathrm{R3SAT}}$, we obtain $\varphi \in \Pi_{\mathrm{NO}}$. On the other hand, since $\Pi$ $\gamma$-avoids $G^{\mathrm{R3SAT}}$, $\varphi \in \Pi_{\mathrm{YES}}$ holds with probability at least $\gamma(n)$ over the choice of a random formula $\varphi \sim \mathcal{D}_n^{\mathrm{R3SAT}}$. □

Since any efficiently computable hitting set generator can be avoided by using a MINKT oracle, as a corollary we obtain Random 3SAT-hardness of MINKT for some $m(n) = \Theta(n)$.

**Corollary 3.16** (Random 3SAT-hardness of MINKT)**.** *Let $\Delta$ be any constant such that $\Delta > 1/\log(8/7) \approx 5.19$. There exists a polynomial-time oracle algorithm that, given oracle access to* MINKT*, solves Random 3SAT of $m(n)$ clauses with probability $1 - 2^{-\Omega(n)}$ for $m(n) = \Delta n$.*

*Proof.* Recall that, by Theorem 2.20, any hitting set generator can be avoided by the set of Kolmogorov-random strings. We thus define $R := \{\, x \in \{0,1\}^* \mid \mathrm{K}_{t(|x|)}(x) > n + \lceil m(n)\log(7\binom{n}{3})\rceil + 3\log n \,\}$, where $t(n)$ is the time it takes to compute $G_n^{\mathrm{R3SAT}}$; note that $R$ is reducible to MINKT via a one-query polynomial-time reduction. Let $\gamma(n)$ be some parameter chosen later. Since $\mathcal{D}_n^{\mathrm{R3SAT}}$ is a uniform distribution, its min-entropy $k(n)$ is $m(n)\log(8\binom{n}{3})$; hence $R$ satisfies the promise of $\Pi$ in Theorem 2.20 if

$$m(n)\log(8\binom{n}{3}) - 1 - \log(1/(1-\gamma(n))) \geq n + \lceil m(n)\log(7\binom{n}{3})\rceil + 3\log n.$$

This inequality is satisfied if $\log(1/(1-\gamma(n))) \leq (\Delta\log(8/7) - 1)n - O(1) = \Omega(n)$. Therefore, $R$ $\gamma$-avoids $G^{\mathrm{R3SAT}}$ as long as $\gamma(n) \leq 1 - 2^{-\Omega(n)}$. Thus by Theorem 3.15, $R$ solves Random 3SAT. □

**Hardness of MKTP**

With some extra work, the Random 3SAT-hardness of MINKT can be extended to the case of MKTP. It is also possible to extend it to a "robust" version of Random 3SAT proposed by Feige [Fei02]: Feige's hypothesis states that there is no polynomial-time algorithm that rejects most formulas, and accepts (not only every satisfiable formula but also) every formula for which all but $\epsilon m$ clauses are satisfiable (henceforth, such a formula is called $(1-\epsilon)$-*satisfiable*).

**Hypothesis 3.17** (Feige [Fei02])**.** *For every fixed $\epsilon > 0$ and for a sufficiently large constant $\Delta$, there is no polynomial-time algorithm that accepts every $(1-\epsilon)$-satisfiable formula, and rejects most formulas.*

Note that the robust version of Random 3SAT is harder than the original version of Random 3SAT (and the original version corresponds to the case when $\epsilon = 0$), and is more robust with respect to how to generate a random 3CNF formula (cf. [Fei02]). Here we show that the robust version can be refuted under an MKTP oracle.

**Theorem 3.18** (Robust Random 3SAT-hardness of MKTP)**.** *There exists a polynomial-time algorithm with oracle access to* MKTP *that refutes Hypothesis 3.17.*

*Proof.* The main idea is that the hitting set generator $G^{\mathrm{R3SAT}}$ of Proposition 3.14 can be modified so that $\mathrm{KT}(G^{\mathrm{R3SAT}}(z))$ is small for every seed $z$, that is, each bit of $G^{\mathrm{R3SAT}}(z)$ is efficiently computable. Moreover, we can modify $G^{\mathrm{R3SAT}}$ so that every $(1-\epsilon)$-satisfiable formula is in the range of $G^{\mathrm{R3SAT}}$. In other words, we will show that, for some appropriately chosen threshold $\theta$, $\mathrm{KT}(\varphi) \leq \theta$ for every $(1-\epsilon)$-satisfiable formula $\varphi$, whereas $\mathrm{KT}(\varphi') > \theta$ with high probability for a random 3CNF formula $\varphi'$. Thus our reduction is a many-one reduction that maps $\varphi$ to $(\varphi, \theta)$.

Define $\theta := m \log(8\binom{n}{3}) - m/2b$ for some constant $b$ specified later. We first claim that $\mathrm{KT}(\varphi) > \theta$ with high probability over the choice of a random 3CNF formula $\varphi$. Indeed, the number of formulas $\varphi$ such that $\mathrm{KT}(\varphi) \leq \theta$ is at most $2^{\theta+1}$ by a simple counting argument (cf. Fact 2.3). Since a random 3CNF formula $\varphi$ is chosen uniformly at random out of the space of cardinality $(2^3\binom{n}{3})^m = 2^{\theta+m/2b}$, the probability that $\mathrm{KT}(\varphi) \leq \theta$ is at most $2^{-m/2b+1}$.

The rest of the proof is devoted to proving every $(1-\epsilon)$-satisfiable formula is of low KT-complexity:

**Claim 3.19.** *Let $\epsilon > 0$ be a sufficiently small constant and $\Delta$ be a sufficiently large constant. Then, for all sufficiently large $n$ and any $(1-\epsilon)$-satisfiable formula $\varphi$ on $n$ variables with $m = \Delta n$ clauses, we have $\mathrm{KT}(\varphi) \leq \theta$.*

In order to claim that the KT-complexity of $\varphi$ is small, we need to implement an efficient procedure that, given an index, outputs the clause of $\varphi$ specified by the index, with random access to a description of $\varphi$. We will describe $\varphi$ by using a $(1-\epsilon)$-satisfying assignment $a \in \{0,1\}^n$, a subset $S \in \binom{[m]}{\epsilon m}$ of the clauses not satisfied by $a$, $(1-\epsilon)m \log(7\binom{n}{3})$ bits to describe the clauses satisfied by $a$, and $\epsilon m \log(8\binom{n}{3})$ bits to describe the clauses not satisfied by $a$.

In order to describe each clause of $\varphi$ efficiently (*i.e.* in time $\mathsf{polylog}(m)$), there are two issues for which we need ideas from succinct data structures. One is an efficient representation of $S$. Information theoretically, $S$ can be described in $\log\binom{m}{\epsilon m} \leq \epsilon m \log(em/\epsilon m) = m\epsilon \log(e/\epsilon)$ bits. However, a naïve representation

of $S$ may not enable us to answer a query $i \stackrel{?}{\in} S$ efficiently; thus, we need the following result.

**Lemma 3.20** (Brodnik and Munro [BM99]). *Let $\epsilon > 0$ be a constant. There exists an algorithm $M$ such that, for every $m \in \mathbb{N}$ and every $S \in \binom{[m]}{\epsilon m}$, there exists a string $d_S$ of length $\log \binom{m}{\epsilon m} + o(\log \binom{m}{\epsilon m})$ such that, given an index $i \in [m]$ and random access to $d_S$, $M$ answers a query $i \stackrel{?}{\in} S$ in time $\mathsf{polylog}(m)$.*

The other issue is the use of the ceiling function (cf. [Pat08, AGvM$^+$18]). Each clause satisfied by $a$ can be described by $\lceil \log(7\binom{n}{3}) \rceil$ bits, which is not necessarily sufficiently smaller than $\log(8\binom{n}{3})$ bits. We thus group consecutive $b$ clauses of $\varphi$ into one block for some constant $b$ so that each block encodes $b$ clauses by using at most $\lceil b \log(7\binom{n}{3}) \rceil$ bits. (In Proposition 3.14, we grouped all the clauses into one block and represented these clauses by using $\lceil m \log(7\binom{n}{3}) \rceil$ bits; however, this representation may not enable us to describe each clause efficiently given random access to the description.) Let $b$ be some universal constant such that $\lceil b \log(8\binom{n}{3}) \rceil - \lceil b \log(7\binom{n}{3}) \rceil \geq 4$ for all large $n \in \mathbb{N}$.

Overall, we can describe a $(1-\epsilon)$-satisfiable formula $\varphi$ by using the following information: (1) A $(1-\epsilon)$-satisfying assignment $a \in \{0,1\}^n$; (2) A string $d_S$ of Lemma 3.20 that represents a subset $S \in \binom{[m]}{\epsilon m}$ of the clauses not satisfied by $a$; (3) $\lceil b \log(7\binom{n}{3}) \rceil$ bits to describe each group of $b$ clauses such that all the clauses in the block are satisfied by $a$; (4) $\lceil b \log(8\binom{n}{3}) \rceil$ bits to describe each group of $b$ clauses such that some clause in the block is not satisfied by $a$. Note that there are $\lceil \frac{m}{b} \rceil$ blocks in total, and there are at most $\epsilon m$ blocks which contain some clause not satisfied by $a$. Given the information, by inspection, one can observe that each clause of $\varphi$ can be described in $\mathsf{polylog}(m)$ time. Thus the KT-complexity of $\varphi$ is at most

$$n + \log \binom{m}{\epsilon m} + o(m) + \left( \left\lceil \frac{m}{b} \right\rceil - \epsilon m \right) \cdot \left\lceil b \log(7\binom{n}{3}) \right\rceil + \epsilon m \cdot \left\lceil b \log(8\binom{n}{3}) \right\rceil .$$

The last two terms can be rewritten as

$$\begin{aligned} &\left\lceil \frac{m}{b} \right\rceil \cdot \left\lceil b \log(8\binom{n}{3}) \right\rceil - \left( \left\lceil \frac{m}{b} \right\rceil - \epsilon m \right) \cdot \left( \left\lceil b \log(8\binom{n}{3}) \right\rceil - \left\lceil b \log(7\binom{n}{3}) \right\rceil \right) \\ &\leq m \log(8\binom{n}{3}) + \frac{m}{b} + o(m) - \frac{m}{2b} \cdot 4, \end{aligned}$$

where, in the last inequality, we take $\epsilon$ small enough so that $\lceil \frac{m}{b} \rceil - \epsilon m \geq \frac{m}{2b}$. Overall, we obtain

$$\begin{aligned} \mathrm{KT}(\varphi) &\leq n + \log \binom{m}{\epsilon m} + m \log(8\binom{n}{3}) - \frac{m}{b} + o(m) \\ &\leq \frac{m}{\Delta} + m\epsilon \log(e/\epsilon) + m \log(8\binom{n}{3}) - \frac{m}{b} + o(m) \\ &\leq m \log(8\binom{n}{3}) - \frac{m}{2b} \leq \theta, \end{aligned}$$

for a sufficiently small $\epsilon > 0$ and a sufficiently large $\Delta$. □

It should be noted that our proof does not seem to carry over to the case of MCSP. The gap between the KT-complexity of satisfiable formulas and random formulas is smaller than $o(|\varphi|)$, and it is not clear how to construct a small

circuit which simulates the random access machine with an additive overhead smaller than $o(|\varphi|)$. We leave as an open question to extend Theorem 3.18 (and Corollary 3.16) to the case of MCSP.

**Open Question 3.21.** Is MCSP Random 3SAT-hard?

### 3.2.3 Hardness of MCSP under Alekhnovich's Hypothesis

While we were not able to prove that MCSP is Random 3SAT-hard, we can refute a strong hypothesis about average-case complexity proposed by Alekhnovich [Ale11] under an MCSP oracle. He considered a problem of solving linear equations under a noise $e$. Let $A$ be an $m \times n$ matrix over GF(2). Let $D_k(A)$ be the distribution of a random vector $Av+e$, where $v$ is a uniform sample from $\mathrm{GF}(2)^n$ and $e \in \mathrm{GF}(2)^n$ is a uniform sample from the vectors of Hamming weight $k$ (*i.e.* the number of ones in $e$ is $k$). Alekhnovich conjectured that there is a matrix such that it is infeasible to distinguish $D_k(A)$ from $D_{k+1}(A)$ efficiently.

**Hypothesis 3.22** (Alekhnovich [Ale11, Conjecture 4.5])**.** *For every $m(n) = \Theta(n)$, there exists a family of $m(n) \times n$ matrices $\{A_n\}_{n\in\mathbb{N}}$ such that, for every function $k(n)$ which satisfies $n^\epsilon < k(n) < n^{1-\epsilon}$ for some constant $\epsilon > 0$, for every efficient algorithm $M$ and every polynomial $p(n)$,*

$$|\Pr[M(D_k(A_n)) = 1] - \Pr[M(D_{k+1}(A_n)) = 1]| \leq 1/p(n),$$

*for all sufficiently large $n$.*

**Theorem 3.23.** *There is a polynomial-time algorithm with oracle access to* MCSP *that refutes Hypothesis 3.22.*

*Proof Sketch.* Alekhnovich showed that Hypothesis 3.22 implies the existence of a cryptographic pseudorandom generator ([Ale11, Lemma 4.14]). On the other hand, by Theorem 3.7, one can invert any auxiliary-input one-way function, and, in particular, one can distinguish the pseudorandom generator from the uniform distribution in polynomial time under an MCSP oracle. (Indeed, a pseudorandom generator $G$ can be distinguished by inverting $G$ and finding a seed of $G$.) □

# Chapter 4

# Non-Black-Box Worst-Case to Average-Case Reductions

In this chapter, we show that the worst-case and average-case complexity of MDLPs are equivalent. As shown in Chapter 3, MDLPs are conjectured to be outside $\mathsf{NP} \cap \mathsf{coNP}$ (or even $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$). On the other hand, there are significant obstacles to establishing an equivalence between the worst-case and average-case hardness of $\mathsf{NP}$: Several results suggest that black-box worst-case to average-case reductions are not likely to be used for reducing any worst-case problem outside $\mathsf{coNP}$ to a distributional $\mathsf{NP}$ problem.

The results presented in this chapter overcomes this barrier. We present the first *non-black-box* worst-case to average-case reduction from a problem conjectured to be outside $\mathsf{coNP}$ to a distributional $\mathsf{NP}$ problem. Specifically, we show that there exists a zero-error randomized polynomial-time algorithm approximating the minimum time-bounded Kolmogorov complexity $\mathrm{K}^t(x)$ within an *additive* error $\widetilde{O}(\sqrt{\mathrm{K}^t(x)})$ if its average-case version admits an errorless heuristic polynomial-time algorithm. We also show that, $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{BPP}$ if and only if its average-case version is easy.

Based on our results, we propose a research program towards excluding Heuristica, i.e., establishing an equivalence between the worst-case and average-case hardness of $\mathsf{NP}$ through the lens of MDLPs.

### Organization

This chapter is organized as follows. After reviewing background in Section 4.1, we state the main results, proof ideas, and our perspective in Section 4.2. In Section 4.3, we present the results about MINKT, and then we present the results about MCSP in Section 4.4.

## 4.1 Background

### 4.1.1 Levin's Average-case Complexity Theory

A traditional complexity class such as $\mathsf{P}$ and $\mathsf{NP}$ measures the performance of an algorithm with respect to the *worst-case* input. However, such a worst-case input may not be found efficiently, and may never be encountered in practice. Average-case complexity, pioneered by Levin [Lev86], aims at analyzing the performance of an algorithm with respect to *random inputs* which can be easily generated by an efficient algorithm. Here we review basic definitions and results on average-case complexity. A survey on average-case complexity can be found in [Imp95, BT06a].

We consider a *distributional problem*, which is a pair of a worst-case problem $L$ and an ensemble of distributions $\mathcal{D} = \{\mathcal{D}_m\}_{m\in\mathbb{N}}$, where $m$ means an instance size and $\mathcal{D}_m$ is a distribution on $\{0,1\}^*$. (We do not require that $\mathcal{D}_m$ is supported on $\{0,1\}^m$; in other words, an instance of size $m$ may not be encoded as a string of length $m$.)

**Definition 4.1** (Distributional Problem)**.** *A pair $(L, \mathcal{D})$ is called a* distributional problem *if $L \subseteq \{0,1\}^*$ and $\mathcal{D} = \{\mathcal{D}_m\}_{m\in\mathbb{N}}$, where each $\mathcal{D}_m$ is a distribution on $\{0,1\}^*$.*

Throughout this chapter, we always use $m$ to denote an instance size.

What kind of distributions should we consider? The original Levin's theory concerns a distribution such that its cumulative probability is computable in polynomial time. The scope of the average-case complexity was later widened by Ben-David, Chor, Goldreich, and Luby [BCGL92] to the case of an *efficiently samplable distribution.*

**Definition 4.2** (Efficiently Samplable)**.** *An ensemble of distributions $\mathcal{D} = \{\mathcal{D}_m\}_{m\in\mathbb{N}}$ is said to be* efficiently samplable *if there exists a randomized polynomial-time machine $M$ such that, on input $1^m$, $M$ outputs a string $x$ according to the distribution $\mathcal{D}_m$ for every $m \in \mathbb{N}$; that is,*

$$\Pr_M[M(1^m) = x_0] = \Pr_{x\sim D_m}[x = x_0]$$

*for every $m \in \mathbb{N}$ and every $x_0 \in \{0,1\}^*$.*

Note that this definition captures the fact that it is important to be able to sample a hard instance efficiently for the purpose of cryptography. In fact, if there is no complexity bound on the distributions, it can be shown that there exists a distribution on which the worst-case and average-case complexity of NP coincide (cf. [LV92]). We thus restrict our attention to efficiently samplable distributions.

**Definition 4.3** (Distributional NP)**.** *An average-case version of* NP *is called a* distributional NP *(denoted by* DistNP*), and is defined as follows:*

$$\mathsf{DistNP} := \{\, (L, \mathcal{D}) \mid L \in \mathsf{NP} \text{ and } \mathcal{D} \text{ is efficiently samplable} \,\}.$$

The performance of an algorithm for a distributional problem $(L, \mathcal{D})$ is measured with respect to the average-case behavior of $A$ on input chosen according to $\mathcal{D}_m$, for each $m \in \mathbb{N}$. There are two possible definitions here: One is the errorless heuristic notion (AvgP), under which an algorithm is not allowed to make any error (in the sense that the algorithm can output "I don't know"). It can be shown that this definition is equivalent to the original notion of Levin (cf. [Imp95, BT06a]), under which the performance of an algorithm is measured with respect to the expected running time over a random input. Another is the heuristic notion (HeurP), under which an algorithm is allowed to make errors. We use the former notion.

**Definition 4.4** (Errorless Heuristic Algorithm)**.** *Let $L \subseteq \{0,1\}^*$ be a language, $\mathcal{D} = \{\mathcal{D}_m\}_{m\in\mathbb{N}}$ be an ensemble of distributions, and $\delta\colon \mathbb{N} \to [0,1]$. We say that an algorithm $A$ is an* errorless heuristic algorithm for $(L, \mathcal{D})$ with failure probability *$\delta$ if*

- *$A(x)$ outputs either $L(x)$ or the special failure symbol $\perp$ for every $x \in \{0,1\}^*$, and*

- $\Pr_{x \sim \mathcal{D}_m}[A(x) = \perp] \leq \delta(m)$ *for every* $m$.

$\mathsf{Avg}_\delta\mathsf{P}$ *denotes the class of distributional problems* $(L, \mathcal{D})$ *such that there exists an errorless heuristic deterministic polynomial-time algorithm for* $(L, \mathcal{D})$ *with failure probability* $\delta$. *Define* $\mathsf{AvgP} := \bigcap_{c \in \mathbb{N}} \mathsf{Avg}_{m^{-c}}\mathsf{P}$.

Similarly, for other complexity classes such as BPP and coNP/poly, the corresponding classes of errorless heuristic algorithms (AvgBPP and Avg-coNP/poly, respectively) can be defined. We defer a formal definition of AvgBPP to Definition 4.35 of Section 4.4.

An example of distributional NP problems is Random 3SAT. By using the terminology reviewed in this section, Random 3SAT is formally defined as the distributional problem $(\mathsf{3SAT}, \mathcal{D}^{\mathrm{R3SAT}})$, where $\mathsf{3SAT} := \{\, \varphi \mid \varphi \text{ is a satisfiable 3CNF formula} \,\}$ and $\mathcal{D}^{\mathrm{R3SAT}}$ is the distribution defined in Definition 3.13 of Subsection 3.2.2.

It is not known whether Random 3SAT is "DistNP-complete". Still, as in the case of the traditional NP-completeness theory, there exists some distributional NP problem complete for DistNP (see, e.g., [Lev86, BCGL92, IL90]). In this sense, we have a problem that is most difficult to solve within DistNP, but the question is: How hard is DistNP itself? Note that if NP = P we trivially have DistNP $\subseteq$ AvgP. The central open question in the average-case complexity theory is whether the converse direction holds, which would conclude that the average-case complexity of NP is essentially the same with the worst-case complexity of NP.

### 4.1.2 Worst-case to Average-case Reductions

In certain settings, such an equivalence between worst-case complexity and average-case complexity is known. We review two of them.

#### For Complexity Classes Beyond the Polynomial-Time Hierarchy

There is a general technique based on error-correcting codes for showing the equivalence between worst-case complexity and average-case complexity. We briefly describe the technique below: Take any "locally-decodable" error-correcting $\mathrm{Enc}\colon \{0,1\}^N \to \{0,1\}^{N^{O(1)}}$ with the following property: There exists a local decoding procedure that, given any index $i \in [N]$ and random access to a string $y$ that is close to $\mathrm{Enc}(x)$, computes $x_i$. Then, given any worst-case problem $f\colon \{0,1\}^n \to \{0,1\}$, by applying the error-correcting code to the truth table of $f$, we can obtain a distributional problem $(\mathtt{fn}(\mathrm{Enc}(\mathtt{tt}(f))), \mathcal{U})$ as hard as the worst-case problem $f$: Indeed, for any heuristic algorithm $M$ solving $\mathtt{fn}(\mathrm{Enc}(\mathtt{tt}(f)))$ on average, $\mathtt{tt}(M)$ can be regarded as a string $y$ that is close to $\mathrm{Enc}(\mathtt{tt}(f))$, and thus by combining $M$ with the local decoding procedure, we obtain another efficient algorithm $M'$ that solves $f$. (The reader is referred to [FF93, BFNW93, STV01] for more details; a nice exposition can be found in the survey of Vadhan [Vad12]). The technique can be applied to complexity classes above the polynomial-time hierarchy such as PSPACE and EXP, since there exists an error-correcting code such that $f \in \mathsf{PSPACE}$ implies $\mathtt{fn}(\mathrm{Enc}(\mathtt{tt}(f))) \in \mathsf{PSPACE}$. Unfortunately, the same technique cannot be applied to any complexity class within the polynomial-time hierarchy [Vio05].

**For Problems within NP ∩ coNP**

Problems based on lattices admit worst-case to average-case reductions from some problems in $\mathsf{NP} \cap \mathsf{coNP}$ to distributional NP problems. In a seminal paper of Ajtai [Ajt96], it is shown that an approximation version of the shortest vector problem of a lattice in $\mathbb{R}^n$ admits a worst-case to average-case reduction. The complexity of approximating the length of a shortest vector depends greatly on an approximation factor. A worst-case to average-case reduction is known when an approximation factor is larger than $\widetilde{O}(n)$ [Mic04, MR07]. Note that Heuristica does not exist if this approximation problem is NP-hard; however, this is unlikely because approximating the length of a shortest vector within a factor of $O(\sqrt{n})$ is in $\mathsf{NP} \cap \mathsf{coNP}$ [GG00, AR05]. Some NP-hardness is known for an approximation factor of $n^{O(1/\log\log n)}$ [HR12].

### 4.1.3 Limits of Worst-Case to Average-Case Reductions within NP

The worst-case to average-case connections mentioned above are all established by means of *reductions*. Namely, in order to show that a distributional problem $(B, \mathcal{D})$ is at least as hard as a worst-case problem $A$, we start with assuming that there is a hypothetical heuristic algorithm $B'$ that solves $B$ on average; then we construct an efficient algorithm (a reduction $R$) that queries to $B'$ and solves $A$. Almost all reduction techniques are "black-box": that is, a proof of the correctness of the reduction $R$ does not rely on the efficiency of $B'$.

It turned out that the power of such a black-box reduction technique is very limited. A line of work showed significant obstacles to establishing worst-case to average-case reductions for NP-complete problems (e.g., [FF93, BT06b, Vio05, AGGM06, BB15] and Chapter 5): Building on the work of Feigenbaum and Fortnow [FF93], Bogdanov and Trevisan [BT06b] showed that if a language $L$ reduces to a distributional NP problem via a black-box nonadaptive randomized polynomial-time reduction, then $L \in \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. Here, the advice "/poly" is mainly used to encode some information about the distributional problem, and can be removed in some cases such as a reduction to inverting one-way functions [AGGM06, BB15] or avoiding hitting set generators (cf. Chapter 5). Therefore, in order to reduce any problem outside $\mathsf{NP} \cap \mathsf{coNP}$ to a distributional NP problem, it is likely that a non-black-box reduction technique is needed.[1]

Gutfreund, Shaltiel and Ta-Shma [GST07] developed a non-black-box technique to show a worst-case to "average-case" reduction; however, the notion of "average-case" is different from the usual one. They showed that, under the assumption that $\mathsf{P} \neq \mathsf{NP}$, for every polynomial-time algorithm $A$ trying to compute SAT, there exists an efficiently samplable distribution $\mathcal{D}_A$ under which $A$ fails to compute SAT on average. The hard distribution $\mathcal{D}_A$ depends on a source code of $A$, and hence it is not necessarily true that there exists a fixed distribution under which SAT is hard on average.

## 4.2 Overview

The main results of this chapter are the first *non-black-box* worst-case to average-case reductions that overcome the limits of black-box reductions. Specifically, we show that approximation versions of MCSP and MINKT are reducible

[1] Here we implicitly used a popular conjecture that $\mathsf{AM} = \mathsf{NP}$ [KvM02]. It should be also noted that an *adaptive* black-box reduction could be used to overcome the barriers.

to their average-case versions. In this section, we state the main results and explain its importance and proof ideas.

### 4.2.1 Results about MCSP

In the case of MCSP, we show that the search version of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is reducible to the distributional $\mathsf{NP}$ problem $(\mathrm{MCSP}[2^{\epsilon n}], \mathcal{U})$ for some constant $\epsilon > 0$. Namely, the distributional problem asks, given as input a random truth table $\mathtt{tt}(f) \sim \{0,1\}^{2^n}$, to decide whether $\mathsf{size}(f) \leq 2^{\epsilon n}$ or not. In fact, it can be shown that the existence of an errorless heuristic algorithm for this problem is equivalent to that of a natural property.

**Theorem 4.5.** *The following are equivalent.*

1. $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in$ Promise-$\mathsf{BPP}$ *for some* $\epsilon > 0$.
2. *There exists a randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *for some* $\epsilon > 0$.
3. $(\mathrm{MCSP}[2^{\epsilon n}], \mathcal{U}) \in \mathsf{AvgBPP}$ *for some constant* $\epsilon \in (0,1)$.
4. *There exists a* $\mathsf{BPP}$*-natural property useful against* $\mathsf{SIZE}(2^{\epsilon n})$.

We note that the approximation factor of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is $2^{(1-\epsilon)n}$, which is only slightly smaller than a trivial approximation factor $2^n$ (indeed, since the maximum circuit size for $n$-variable functions is at most $2^n$, there exists a trivial polynomial-time algorithm that solves $\mathrm{Gap}_0\mathrm{MCSP}$ unconditionally).

### 4.2.2 Results about MINKT

While the quality of the approximation for MCSP is quite bad, we are able to obtain a much better approximation in the case of MINKT. We consider a problem of deciding whether $x$ is $r$-nonrandom string with respect to $\mathrm{K}_t$, where a random instance $(x, 1^t)$ of size $m \in \mathbb{N}$ is chosen by sampling $t \sim [m]$ and $x \sim \{0,1\}^{m-t}$. More formally, we consider a parameterized version of MINKT such that the threshold of randomness is fixed to $r\colon \mathbb{N} \to \mathbb{N}$.

**Definition 4.6** (Parameterized MINKT)**.** *For a function* $r\colon \mathbb{N} \to \mathbb{N}$*, define*

$$\mathrm{MINKT}[r] := \{\, (x, 1^t) \mid \mathrm{K}_t(x) < r(|x|) \,\}.$$

We consider the following distribution on a pair $(x, 1^t)$ of a binary string and a unary string.

**Definition 4.7** (Uniform distribution with auxiliary unary input)**.** *Define an ensemble of distributions* $\mathcal{D}^{\mathrm{KT}} := \{\mathcal{D}^{\mathrm{KT}}_m\}_{m\in\mathbb{N}}$*, where, for each* $m \in \mathbb{N}$*,* $\mathcal{D}^{\mathrm{KT}}_m$ *is defined as the output distribution of the following algorithm: Pick* $t \sim [m]$ *and* $x \sim \{0,1\}^{m-t}$ *randomly. Output* $(x, 1^t)$.

By this definition, it is obvious that $\mathcal{D}^{\mathrm{KT}}$ is efficiently samplable.[2] Therefore:

**Fact 4.8.** $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{DistNP}$ *if* $r\colon \mathbb{N} \to \mathbb{N}$ *is efficiently computable.*

[2] A minor detail is that a randomized machine cannot sample $t \sim [m]$ exactly when $m$ is not a power of 2, because a coin flip of a randomized machine is usually defined as a binary string. However, it is possible to efficiently sample a distribution exponentially close to $\mathcal{D}^{\mathrm{KT}}$, and thus this detail can be safely ignored with an exponentially small error.

The main technical result of this chapter is that if there exists a polynomial-time errorless heuristic algorithm $A$ that solves $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$ for $r(n) \approx n$, then one can compress any given string $x$ into an efficient program $M_x$ of almost shortest length $\mathrm{K}_t(x) + \widetilde{O}\big(\sqrt{\mathrm{K}_t(x)}\big)$ for *every* input $(x, 1^t)$ in polynomial time. The reason why our reduction is non-black-box is that the efficient program $M_x$ incorporates the heuristic algorithm $A$. In particular, when $A$ is not an efficient algorithm, there is no guarantee that the output $M_x$ of the compression algorithm is an efficient program. We state it more formally:

**Theorem 4.9** (Compression Under $\mathsf{DistNP} \subseteq \mathsf{AvgP}$)**.** *Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that for some constant $c > 0$, for all large $n \in \mathbb{N}$, $n - c\sqrt{n}\log n \leq r(n) < n$. Assume that $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{1/6m}\mathsf{P}$. Then, there exists a zero-error randomized polynomial-time algorithm that, on input $(x, 1^t)$, outputs a program $M$ of size $\leq \mathrm{K}_t(x) + O\big((\log|x|)\sqrt{\mathrm{K}_t(x)} + (\log|x|)^2\big)$ such that $M$ outputs $x$ in $\mathsf{poly}(|x|, t)$ steps.*

It should be noted that the compression algorithm works for *every* input $(x, 1^t)$. As a corollary of Theorem 4.9, an approximation version of MINKT admits a $\mathsf{ZPP}$ algorithm under the same assumption:

**Corollary 4.10.** *Under the same assumption with Theorem 4.9, there exists some $\sigma(n, s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ and some polynomial $\tau(n, t)$ such that $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in$ Promise-$\mathsf{ZPP}$.*

We note that the approximation error $\sigma$ is so small that the Random 3SAT-hardness holds for $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. Indeed, one can easily see that the proof of Random 3SAT-hardness of MINKT (Corollary 3.16) also works for $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ for any $\sigma$ such that $\sigma(n, s) \leq s + o(s/\log s) + o(n/\log n)$. Moreover, our proof can be seen as a non-black-box *nonadaptive* reduction from $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ to $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$, and thus our proof cannot be made black-box unless $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP/poly}$ (and in particular unless Random3SAT $\in$ $\mathsf{Avg}$-$\mathsf{coNP/poly}$): indeed, otherwise by the result of Bogdanov and Trevisan [BT06b], we would obtain $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in \mathsf{coNP/poly}$.

### 4.2.3 Perspective: An Approach Towards Excluding Heuristica

We propose a research program towards excluding Heuristica through the lens of MCSP or MINKT. Note that if $\mathsf{NP} \leq_T^{\mathsf{BPP}} \mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ then we obtain the following by Theorem 4.9: If $\mathsf{NP} \not\subseteq \mathsf{BPP}$ then $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$, which means that Heuristica does not exist.

Unfortunately, there are still several obstacles we need to overcome in order for this research program to be completed. Although our proofs overcome the limits of black-box reductions, our proofs do *relativize*. And there is a relativization barrier for excluding Heuristica: Impagliazzo [Imp11] constructed an oracle $A$ such that $\mathsf{DistNP}^A \subseteq \mathsf{AvgP}^A$ and $\mathsf{NP}^A \cap \mathsf{coNP}^A \not\subseteq \mathsf{P}^A/\mathsf{poly}$. Under the same oracle, it follows from a relativized version of Theorem 4.9 that $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}^A$ is not $\mathsf{NP}^A$-hard under $\mathsf{P}^A/\mathsf{poly}$-Turing reductions. Thus it requires some nonrelativizing technique to establish $\mathsf{NP}$-hardness of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ even under $\mathsf{P/poly}$-Turing reductions. (Previously, Ko [Ko91] constructed a relativized world where MINKT is not $\mathsf{NP}$-hard under $\mathsf{P}$-Turing reductions.)

We also mention that there are a number of results showing that proving $\mathsf{NP}$-hardness of MCSP is extremely difficult or impossible under reducibility notions stronger than $\mathsf{P/poly}$-Turing reductions. However, few is known for weaker reducibility notions such as $\mathsf{NP} \cap \mathsf{coNP}$ reductions. We conjecture that the following is a feasible research question.

**Open Question 4.11.** Prove the following (or explain why it is difficult to resolve): Let $\sigma, \tau$ be arbitrary parameters as in Theorem 4.9. $\mathrm{Gap}_{\sigma,\tau}$MINKT is NP-hard under coNP/poly-Turing reductions. That is, $\mathsf{NP} \subseteq \mathsf{coNP}^A/\mathsf{poly}$ for any oracle $A$ that satisfies the promise of $\mathrm{Gap}_{\sigma,\tau}$MINKT.

Note that the choice of reducibility is somewhat subtle: The relativization barrier applies to P/poly reductions, but it is not known whether a similar barrier applies to coNP/poly reductions. Ko [Ko91] also speculated that MINKT might be NP-complete under $\mathsf{NP} \cap \mathsf{coNP}$ reductions. We mention that there is nonrelativizing proof techniques to prove PSPACE-completeness of a space-bounded version of MINKT under ZPP-Turing reductions and EXP-completeness of an exponential-time version of MINKT under $\mathsf{NP} \cap \mathsf{coNP}$-Turing reductions (cf. [ABK+06b]).

A positive answer to Open Question 4.11 implies the following: If $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, then $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$. This will base the hardness of DistNP on a plausible worst-case assumption of NP, and in particular, an assumption that the polynomial-time hierarchy does not collapse. Currently, no worst-case hardness assumption on the polynomial-time hierarchy is known to imply $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$.

### 4.2.4 Proof Overview

Our starting point is the Nisan-Wigderson generator [NW94]. They presented a (complexity-theoretic) pseudorandom generator $\mathrm{NW}^f$ secure against small circuits, based on any "hard" function $f$ (in the sense that $f$ cannot be approximated by small circuits, that is, $\Pr_x[f(x) = C(x)] \leq \frac{1}{2} + \epsilon$ for some small $\epsilon > 0$ and any small circuit $C$).

Its security is proved by the following reduction: Given any statistical test $T$ that distinguishes the output distribution of $\mathrm{NW}^f$ from the uniform distribution, one can construct a small $T$-oracle circuit $C^T$ that approximates $f$. If $T$ can be implemented by a small circuit, then this is a contradiction to the assumption that $f$ is hard; thus the pseudorandom generator is secure. Such a security proof turns out to be quite fruitful not only for derandomization [KvM02, IW01, TV07], but also for Trevisan's extractor [Tre01a], investigating the power of Kolmogorov-random strings [ABK+06b], and the generic connection between learning and natural proof [CIKK16].

Our proofs also make use of a security proof. It enables us to transform any statistical test $T$ for $\mathrm{NW}^f$ to a small circuit $C^T$ that describes a $(\frac{1}{2}+\epsilon)$-fraction of the truth table of $f$. Moreover, as observed in [IW01], such small circuits can be constructed efficiently. By using a list-decodable error-correcting code Enc, given any statistical test $T$ for $\mathrm{NW}^{\mathrm{Enc}(x)}$, one can efficiently find a short description for $x$ under the oracle $T$.

We argue that there is a statistical test $T$ for $\mathrm{NW}^{\mathrm{Enc}(x)}$ under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. Consider the distributional NP problem $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$. A crucial observation is that there are few nonrandom strings (i.e., compressible by a short program); that is, there are few YES instances in MINKT$[r]$. Thus any errorless heuristic algorithm solving $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$ must reject a large fraction of random strings. This gives rise to a dense subset $T \in P$ of random strings, and it can be shown that $T$ is a statistical test for any hitting set generator.

As a consequence, we obtain an efficient algorithm that, on input $x$, outputs a short program $d$ describing $x$ under the oracle $T$. Since $T$ can be accepted by some polynomial-time algorithm (that comes from the errorless heuristic algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$), we can describe $x$ by using the description $d$ and *a source code* of the algorithm accepting $T$. This is the crucial part in which our proof is

non-black-box; we need a source code of the errorless heuristic algorithm in order to have a short description for $x$. We then obtain a randomized polynomial-time search algorithm for $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$.

The proof sketch above enables us to find a somewhat short description, but it is not sufficient to obtain a description of length $(1+o(1))\cdot \mathrm{K}_t(x)$, nor to obtain the Random 3SAT-hardness of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. To optimize the quality of the approximation, we need to exploit an improvement of the Nisan-Wigderson generator (and Trevisan's extractor), given by Raz, Reingold and Vadhan [RRV02].

Finally, the randomized algorithm described above can be made zero-error; indeed, if $\mathsf{DistNP} \subseteq \mathsf{AvgZPP}$, then any randomized algorithm can be made zero-error (as mentioned in [Imp95] without a proof). This is because a Kolmogorov-random string $w$ can be found by picking a string uniformly at random, and one can check whether $w$ is Kolmogorov-random or not by using an errorless heuristic algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$; by using $w$ as a source of a hard function and invoking the hardness versus randomness framework again, we can derandomize the rest of the randomized computation. (The zero-error algorithm may fail only if no Kolmogorov-random string is found.)

Interestingly, we invoke the hardness versus randomness framework *twice* for completely different purposes. On one hand, to derandomize a randomized computation, it is desirable to minimize the seed length of a pseudorandom generator, because we need to exhaustively search all the seeds. On the other hand, to obtain a short description, it is desirable to minimize the *output* length of a pseudorandom generator (or, in other words, to maximize the seed length); this is because the efficiency of the security proof is dominated by the output length.

To prove a similar equivalence between worst-case and average-case hardness of MCSP, there is one difficulty: An error-correcting code Enc may significantly increase the circuit complexity of $f$. As a consequence, for a function $f$ that can be computed by a small circuit, the circuit complexity of the output of $\mathrm{NW}^{\mathrm{Enc}(f)}$ is not necessarily small, and thus an errorless heuristic algorithm for MCSP may not induce a statistical test for $\mathrm{NW}^{\mathrm{Enc}(f)}$; here, the circuit complexity of a string $x$ refers to the size of a smallest circuit whose truth table is $x$. Nevertheless, it is still possible to amplify the hardness of $f$ while preserving the circuit complexity of $f$. Indeed, Carmosino, Impagliazzo, Kabanets, and Kolokolova [CIKK16] established a generic reduction from approximately learning to natural properties, by using the fact that a natural property is a statistical test for $\mathrm{NW}^{\mathrm{Amp}(f)}$, where $\mathrm{Amp}(f)$ denotes a hardness amplified version of $f$. We observe that their approximately learning is enough to achieve the approximation factor stated in Theorem 4.5. Moreover, as observed by Hirahara and Santhanam [HS17], a natural property is essentially an errorless heuristic algorithm for MCSP. By combining these results, we obtain a search to average-case reduction for GapMCSP.

## 4.3 Worst-Case to Average-Case Reduction for MINKT

### 4.3.1 Preliminaries

We first introduce several notations. To explain a consequence of the security proof of the Nisan-Wigderson generator, it is convenient to introduce an approximation version of Kolmogorov complexity.

**Definition 4.12** (Approximation version of Time-bounded Kolmogorov complexity)**.** *For functions $f, g\colon \{0,1\}^\ell \to \{0,1\}$, define $\mathrm{dist}(f,g) := \Pr_{x\sim\{0,1\}^\ell}[f(x) \neq g(x)]$. For a function $f\colon \{0,1\}^\ell \to \{0,1\}$, an integer $t \in \mathbb{N}$, and an oracle*

$A \subseteq \{0,1\}^*$, *define* $\mathrm{K}^A_{t,\delta}(f)$ *as the minimum length of a string* $d$ *such that* $U^A(d)$ *outputs* $\mathtt{tt}(g)$ *of length* $2^\ell$ *within* $t$ *steps and* $\mathrm{dist}(f,g) \le 1/2 - \delta$.

We next introduce the notation for a certificate for MINKT. Note that since MINKT $\in \mathsf{NP}$, every YES instance has a certificate whose correctness can be checked efficiently.

**Definition 4.13** (Certificate for MINKT)**.** *For an oracle* $A \subseteq \{0,1\}^*$, *integers* $s,t \in \mathbb{N}$, *and a string* $x \in \{0,1\}^*$, *a string* $d \in \{0,1\}^*$ *is called a* certificate for $\mathrm{K}^A_t(x) \preceq s$ *if* $U^A(d)$ *outputs* $x$ *within* $t$ *steps and* $|d| \le s$. *A certificate for* $\mathrm{K}^A_{t,\delta}(x) \preceq s$ *is defined in a similar way.*

In this terminology, for proving Theorem 4.9, on input $(x, 1^t)$, we seek a certificate for

$$\mathrm{K}_{t'}(x) \preceq \mathrm{K}_t(x) + O\big((\log|x|)\sqrt{\mathrm{K}_t(x)} + (\log|x|)^2\big)$$

for some $t' = \mathsf{poly}(|x|,t)$. Note here that "$\preceq$" is just a symbol, and "$\mathrm{K}_t(x) \preceq s$" should be interpreted as a tuple $(x, 1^t, 1^s)$, which is an instance of MINKT.

Theorem 4.9 can be interpreted as solving a search version of $\mathrm{Gap}_{\sigma,\tau}$MINKT. We formally define the search problem associated with $\mathrm{Gap}_{\sigma,\tau}$MINKT below.

**Definition 4.14.** *The search version of* $\mathrm{Gap}_{\sigma,\tau}$MINKT *is defined as follows.*

- *Input: A string* $x \in \{0,1\}^*$ *and an integer* $t \in \mathbb{N}$ *represented in unary.*
- *Output: A certificate for* $\mathrm{K}_{t'}(x) \preceq \sigma(|x|, \mathrm{K}_t(x))$ *for any* $t' \ge \tau(|x|,t)$.

*A randomized algorithm* $A$ *is called a* zero-error *randomized algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}$MINKT *if, for every* $x \in \{0,1\}^*$ *and* $t \in \mathbb{N}$, $A(x,1^t)$ *outputs a certificate for* $\mathrm{K}_{t'}(x) \preceq \sigma(|x|, \mathrm{K}_t(x))$ *whenever* $A(x,1^t) \neq \perp$, *and* $A(x,1^t)$ *outputs* $\perp$ *with probability at most* $\frac{1}{2}$.

We observe that this is indeed "the" search version of $\mathrm{Gap}_{\sigma,\tau}$MINKT by showing that the decision version is easier than its search version:

**Fact 4.15** (Decision reduces to search)**.** *Let* $\sigma, \tau \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *be any efficiently computable and nondecreasing functions. If there exists a zero-error randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}$MINKT, *then* $\mathrm{Gap}_{\sigma,\tau}$MINKT $\in$ Promise-$\mathsf{ZPP}$.

*Proof.* The main point is that the zero-error randomized search algorithm does not err in the sense that it outputs an approximately shortest certificate whenever it succeeds. Therefore, given a zero-error randomized algorithm $M$ solving the search version of $\mathrm{Gap}_{\sigma,\tau}$MINKT, the following algorithm solves the decision version: On input $(x, 1^t, 1^s)$, run $M$ on input $(x, 1^t)$. If $M$ outputs $\perp$, then output $\perp$ and halt. Otherwise, $M$ outputs some certificate $d$. Accept iff $|d| \le \sigma(|x|, s)$ and $U(d)$ outputs $x$ in $\tau(|x|,t)$ steps.

We claim the correctness of this algorithm. If $(x, 1^t, 1^s)$ is a YES instance of $\mathrm{Gap}_{\sigma,\tau}$MINKT, then we obtain a certificate $d$ for $\mathrm{K}_{t'}(x) \preceq \sigma(|x|, \mathrm{K}_t(x))$ where $t' := \tau(|x|,t)$ unless $M$ outputs $\perp$; that is, $U(d)$ outputs $x$ in $t'$ steps, and $|d| \le \sigma(|x|, \mathrm{K}_t(x)) \le \sigma(|x|, s)$. Thus the algorithm above accepts with probability at least $\frac{1}{2}$. If $(x, 1^t, 1^s)$ is a NO instance of $\mathrm{Gap}_{\sigma,\tau}$MINKT, then we have $\mathrm{K}_{t'}(x) > \sigma(|x|, s)$ for $t' := \tau(|x|,t)$. Thus the algorithm rejects unless $M$ outputs $\perp$. □

The following is a simple but crucial lemma in which our proof becomes non-black-box:

**Lemma 4.16.** *Let $T \in \mathsf{P}$. Then there exists some polynomial $p$ such that $\mathrm{K}_{t'}(x) \leq \mathrm{K}^T_t(x) + O(1)$ for any $x \in \{0,1\}^*$ and any $t, t'$ such that $t' \geq p(t)$. Moreover, given a certificate for $\mathrm{K}^T_t(x) \preceq s$, one can efficiently find a certificate for $\mathrm{K}_{t'}(x) \preceq s + O(1)$.*

We will use this lemma for an errorless heuristic polynomial-time algorithm accepting $T$ (in Theorem 4.9). Thus, the output of our non-black-box reduction will be a certificate for $\mathrm{K}_{t'}(x)$ which incorporates a source code of the errorless heuristic polynomial-time algorithm.

*Proof.* Let $M_0$ be a polynomial-time machine that accepts $T$. Consider the following machine $M$: On input $d \in \{0,1\}^*$, simulate $U^T(d)$ using $M_0$; that is, if $U$ makes a query $q$ to the oracle $T$, then run $M_0$ on input $q$ and answer the query $q$ with $M_0(q)$. Then $M$ outputs what $U^T(d)$ outputs.

Now suppose that $U^T(d)$ outputs $x$ in $t$ steps. Then, by the definition, $M(d)$ outputs $x$ in $p_0(t)$ steps for some polynomial $p_0$ (that depends only on the running time of $M_0$); thus, $U(M,d)$ outputs $x$ in $p_U(p_0(t))$ steps, where $p_U$ is the slowdown of the universal Turing machine. Hence we obtain $\mathrm{K}_{t'}(x) \leq \mathrm{K}^T_t(x) + O(|M|)$ for $t' \geq p(t) := p_U(p_0(t))$.

To see the "moreover" part, given a certificate $d$ for $\mathrm{K}^T_t(x) \preceq s$, we may simply output $(M,d)$ as a certificate for $\mathrm{K}_{t'}(x) \preceq s + O(|M|)$. □

### 4.3.2 Short Certificate Under a Dense Subset of Random Strings

In this subsection, we present an efficient algorithm that outputs a certificate for GapMINKT, given an oracle that accepts some dense subset of random strings. The existence of such an oracle will be justified in the next subsection under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$. We introduce the notation for $r$-random strings.

**Definition 4.17** ($r$-random strings)**.** *Let $R_t[r]$ denote the set of all $r$-random strings with respect to $\mathrm{K}_t$; that is, $R_t[r] := \{\, x \in \{0,1\}^* \mid \mathrm{K}_t(x) \geq r(|x|) \,\}$.*

In particular, a set $A \subseteq \{0,1\}^m$ is said to be a $\delta$-dense subset of $r$-random strings $R_t[r]$ if $A \subseteq R_t[r]$ and $|A| \geq 2^m\delta$.

The main idea is that a dense subset of random strings gives rise to a statistical test distinguishing any pseudorandom generator from the uniform distribution. Indeed, take any efficiently computable function $G\colon \{0,1\}^d \to \{0,1\}^m$ where $d \lesssim r(m)$; then any range $G(z)$ of $G$ can be described by its seed $z$ in polynomial time; hence $G(z)$ is not $r$-random since $\mathrm{K}_t(G(z)) \lesssim d \lesssim r(m)$; thus a $\delta$-dense subset $T$ of $r$-random strings is a *statistical test* for $G$ with advantage $\delta$, i.e., $\left|\Pr_{w\sim\{0,1\}^m}[w \in T] - \Pr_{z\sim\{0,1\}^d}[G(z) \in T]\right| \geq \delta$. We will use this fact to break the Nisan-Wigderson generator.

We proceed to define the Nisan-Wigderson generator $\mathrm{NW}^f$. Originally, Nisan and Wigderson [NW94] defined the notion of *design* as a family of subsets $S_1, \ldots S_m$ such that $|S_i \cap S_j|$ is small for every distinct $i, j \in [m]$. As observed by Raz, Reingold and Vadhan [RRV02], a weaker notion is sufficient for a security proof of the Nisan-Wigderson generator. Our notion is, however, different from the weak design defined in [RRV02] due to some technical details.

**Definition 4.18.** *We say that a family $\mathcal{S} = (S_1, \ldots, S_m)$ of subsets of $[d]$ is a* $(\ell, \rho)$-design *if $|S_i| = \ell$ and $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} + m - i \leq \rho m$ for every $i \in [m]$.*

There is an efficient way to construct such a family with nice parameters.

**Lemma 4.19** (follows from [RRV02, Lemma 15])**.** *For any $m, \ell, d \in \mathbb{N}$ such that $d/\ell \in \mathbb{N}$, there exists a $(\ell, \exp(\ell^2/d))$-design $\mathcal{S}_{m,\ell,d} = (S_1, \ldots, S_m) \subseteq \binom{[d]}{\ell}$. Moreover, the family $\mathcal{S}_{m,\ell,d}$ can be constructed by a deterministic algorithm in time $\mathsf{poly}(m, d)$.*

*Proof Sketch.* Raz, Reingold and Vadhan [RRV02] showed how to construct, in time $\mathsf{poly}(m, d)$, a family of subsets $S_1, \ldots, S_m \subseteq [d]$ of size $\ell$ such that $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} \leq (1 + \ell/d)^\ell \cdot (i-1) \leq \exp(\ell^2/d) \cdot i$ for every $i \in [m]$. (The family is constructed by dividing $[d]$ into $\ell$ disjoint blocks of size $d/\ell$, and, for each $i \in [m]$, choosing one random element out of each block and adding it to $S_i$. The construction can be derandomized by the method of conditional expectations.) The same family satisfies the condition that $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|} + m - i \leq \exp(\ell^2/d) \cdot m$ for every $i \in [m]$. □

For a string $z \in \{0,1\}^d$ and a subset $S = \{i_1 < \cdots < i_\ell\} \subseteq [d]$, we denote by $z_S \in \{0,1\}^\ell$ the string $z_{i_1} \cdots z_{i_\ell}$. To avoid introducing a new variable, we treat $d/\ell$ as if it is a variable.

**Definition 4.20** (Nisan-Wigderson generator [NW94])**.** *For a function $f\colon \{0,1\}^\ell \to \{0,1\}$ and parameters $m, \ell, d/\ell \in \mathbb{N}$, define the* Nisan-Wigderson generator $\mathrm{NW}^f_{m,d}\colon \{0,1\}^d \to \{0,1\}^m$ *as $\mathrm{NW}^f_{m,d}(z) := f(z_{S_1}) \cdots f(z_{S_m})$ for every $z \in \{0,1\}^d$, where $(S_1, \ldots, S_m) := \mathcal{S}_{m,\ell,d}$.*

Nisan and Wigderson [NW94] showed that if $f$ is a hard function (i.e. $f$ cannot be approximated by small circuits) then $\mathrm{NW}^f_{m,d}$ is a pseudorandom generator secure against small circuits. The security proof of the Nisan-Wigderson generator transforms any statistical test for $\mathrm{NW}^f_{m,d}$ into a small circuit that approximately describes $f$. Moreover, as observed in [IW01], such small circuits can be constructed efficiently. We now make use of these facts to obtain a short description for $f$. Our proof is similar to the construction of Trevisan's extractor [Tre01a], but we need to argue the efficiency.

**Lemma 4.21.** *There exist some polynomial* $\mathsf{poly}$ *and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A function $f\colon \{0,1\}^\ell \to \{0,1\}$ represented as its truth table, parameters $m, d/\ell, \delta^{-1} \in \mathbb{N}$ represented in unary, and oracle access to $T \subseteq \{0,1\}^m$.*

Promise: *We assume that the oracle $T$ is a statistical test for $\mathrm{NW}^f_{m,d}$ with advantage $\delta$. That is,*

$$\left| \Pr_{z \sim \{0,1\}^d} \left[ T(\mathrm{NW}^f_{m,d}(z)) = 1 \right] - \Pr_{w \sim \{0,1\}^m} \left[ T(w) = 1 \right] \right| \geq \delta. \tag{4.1}$$

Output: *A certificate for $\mathrm{K}^T_{t,\delta/2m}(f) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(md))$, for any $t \geq \mathsf{poly}(m, d, 2^\ell)$.*

*Proof.* We first prove $\mathrm{K}^T_{t,\delta/m}(f) \leq \exp(\ell^2/d) \cdot m + d + O(\log(md))$. We will then explain how to obtain a certificate efficiently (with the small loss in the quality $\delta/m$ of the approximation).

The first part is proved by a standard hybrid argument as in [NW94]. Without loss of generality, we may ignore the absolute value of (4.1); more precisely, let $T_b(w) := T(w) \oplus b$ for some $b \in \{0,1\}$ so that $\mathbb{E}_{z,w} \left[ T_b(\mathrm{NW}^f_{m,d}(z)) - T_b(w) \right] \geq \delta$. For every $i \in [m]$, define a hybrid distribution $H_i := f(z_{S_1}) \cdots f(z_{S_i}) \cdot w_{i+1} \cdots w_m$

for $z \sim \{0,1\}^d$ and $w \sim \{0,1\}^m$. As $H_0$ and $H_m$ are distributed identically to $w \sim \{0,1\}^m$ and $\mathrm{NW}^f_{m,d}(z)$ for $z \sim \{0,1\}^d$, respectively, we have $\mathbb{E}[T_b(H_m) - T_b(H_0)] \geq \delta$. Pick $i \sim [m]$ uniformly at random. Then we obtain $\mathbb{E}_i[T_b(H_i) - T_b(H_{i-1})] \geq \delta/m$.

We can exploit this advantage to predict the next bit of the PRG (due to Yao [Yao82]; a nice exposition can be found in [Vad12, Proposition 7.16]). For each fixed $i \in [m]$, $c \in \{0,1\}$, $w_{[m]\setminus[i]} \in \{0,1\}^{m-i}$, and $z_{[d]\setminus S_i} \in \{0,1\}^{d-\ell}$, consider the following circuit $P^{T_b}$ for predicting $f$: On input $x \in \{0,1\}^\ell$, set $z_{S_i} := x$ and construct $z \in \{0,1\}^d$. Output $T_b(f(z_{S_1}) \cdots f(z_{S_{i-1}}) \cdot c \cdot w_{i+1} \cdots w_m) \oplus c \oplus 1$. A basic idea here is that if $c = f(z_{S_i})$ $(= f(x))$ then the input distribution of $T_b$ is identical to $H_i$ and thus $T_b$ is likely to output 1, in which case we should output $c$ for predicting $f$. By a simple calculation, it can be shown that $\Pr[P^{T_b}(x) = f(x)] \geq \frac{1}{2} + \frac{\delta}{m}$, where the probability is taken over all $i \sim [m]$, $c \sim \{0,1\}$, $w_{[m]\setminus[i]} \sim \{0,1\}^{m-i}$, $z_{[d]\setminus S_i} \sim \{0,1\}^{d-\ell}$, and $x \sim \{0,1\}^\ell$. In particular, by averaging, there exists some $i, c, w_{[m]\setminus[i]}, z_{[d]\setminus S_i}$ such that $\Pr_{x\sim\{0,1\}^\ell}\left[P^{T_b}(x) = f(x)\right] \geq \frac{1}{2} + \frac{\delta}{m}$.

Therefore, it is sufficient to claim that the circuit $P$ has a small description. Note that the value of $f$ needed in the computation of $P$ can be hardwired into the circuit using $\sum_{j<i} 2^{|S_i \cap S_j|}$ bits. Given oracle access to $T$, we can describe the $(\frac{1}{2} + \frac{\delta}{m})$-fraction of the truth table of $f$ by specifying $m, \ell, d, b, c, i, w_{[m]\setminus[i]}, z_{[d]\setminus S_i}$, and the hardwired table of the values of $f$. This procedure takes time roughly $\mathsf{poly}(m,d) + \mathsf{poly}(2^\ell)$ (for computing the design and evaluating the entire truth table of $P^{T_b}$). The length of the description is at most $\sum_{j<i} 2^{|S_i \cap S_j|} + (m - i) + (d - \ell) + O(\log(md)) \leq \exp(\ell^2/d) \cdot m + d + O(\log(md))$. Thus we have $\mathrm{K}^T_{t,\,\delta/m}(f) \leq \exp(\ell^2/d) \cdot m + d + O(\log(md))$.

To find a certificate efficiently, observe that a random choice of $(c, i, w_{[m]\setminus[i]}, z_{[d]\setminus S_i})$ is sufficient in order for the argument above to work. That is, pick $c \sim \{0,1\}$, $i \sim [m]$, $w_{[m]\setminus[i]} \sim \{0,1\}^{m-i}$, and $z_{[d]\setminus S_i} \sim \{0,1\}^{d-\ell}$. Then a Markov style argument shows that, with probability at least $\delta/2m$, we obtain $\Pr_{x\sim\{0,1\}^\ell}\left[P^{T_b}(x) = f(x)\right] \geq \frac{1}{2} + \frac{\delta}{2m}$. By trying each $b \in \{0,1\}$ and trying the random choice $O(m/\delta)$ times, we can find at least one certificate for $\mathrm{K}^T_{t,\delta/2m}(f)$ with high probability. □

We will update Lemma 4.21 by incorporating a list-decodable error-correcting code, so that we obtain a certificate for $\mathrm{K}^T_t(x)$ instead of $\mathrm{K}^T_{t,\delta/2m}(f)$.

**Definition 4.22** (List-decodable error-correcting code; cf. [Vad12])**.** *For every $n, m, L \in \mathbb{N}$ and $\epsilon > 0$, a function* $\mathrm{Enc}\colon \{0,1\}^n \to \{0,1\}^m$ *is called a* $(L, \frac{1}{2} - \epsilon)$-list-decodable error-correcting code *if there exists a function* $\mathrm{Dec}\colon \{0,1\}^m \to (\{0,1\}^n)^L$ *such that, for every $x \in \{0,1\}^n$ and $r \in \{0,1\}^m$ with* $\mathrm{dist}(\mathrm{Enc}(x), r) \leq \frac{1}{2} - \epsilon$*, we have $x \in \mathrm{Dec}(r)$. We call* Dec *a* list decoder *of* Enc.

For our purpose, it is sufficient to use any standard list-decodable code such as the concatenation of a Reed-Solomon code and an Hadamard code.

**Theorem 4.23** (see, e.g., [STV01] and [Vad12, Problem 5.2])**.** *For any $n \in \mathbb{N}$ and $\epsilon > 0$, there exists a function* $\mathrm{Enc}_{n,\epsilon}\colon \{0,1\}^n \to \{0,1\}^{2^\ell}$ *with $\ell = O(\log(n/\epsilon))$ that is a $(\mathsf{poly}(1/\epsilon), \frac{1}{2} - \epsilon)$-list-decodable error-correcting code. Moreover,* $\mathrm{Enc}_{n,\epsilon}$ *and its list decoder* $\mathrm{Dec}_{n,\epsilon}$ *are computable in time* $\mathsf{poly}(n, 1/\epsilon)$.

In what follows, we implicitly regard a string $\mathrm{Enc}_{n,\epsilon}(x) \in \{0,1\}^{2^\ell}$ of length $2^\ell$ as a function on $\ell$-bit inputs.

**Corollary 4.24.** $\mathrm{K}^A_{t'}(x) \leq \mathrm{K}^A_{t,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x)) + O(\log(n/\epsilon))$ *for any string* $x \in \{0,1\}^*$*, any oracle* $A$*, and any* $t' \geq t + \mathsf{poly}(n, 1/\epsilon)$*. Moreover, given any* $x$ *and any certificate for* $\mathrm{K}^A_{t,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x)) \preceq s$*, one can find a certificate for* $\mathrm{K}^A_{t'}(x) \preceq s + O(\log(n/\epsilon))$ *in time* $t + \mathsf{poly}(n, 1/\epsilon)$ *with oracle access to* $A$*.*

*Proof.* Consider the following procedure $M$: Given input $d_0 \in \{0,1\}^*$ and $n, \epsilon^{-1} \in \mathbb{N}$ and index $i \in \mathbb{N}$, output the $i$th string of $\mathrm{Dec}_{n,\epsilon}(U^A(d_0))$. By the definition, there exists some description $d_0$ of length $\mathrm{K}^A_{t,\epsilon}(\mathrm{Enc}_{n,\epsilon}(x))$ such that $U^A(d_0)$ outputs some string $r$ within time $t$ and $\mathrm{dist}(\mathrm{Enc}_{n,\epsilon}(x), r) \leq \frac{1}{2} - \epsilon$. Thus there exists an index $i \leq \mathsf{poly}(1/\epsilon)$ such that the $i$th string of $\mathrm{Dec}_{n,\epsilon}(r)$ is equal to $x$. Hence, we obtain $\mathrm{K}^A_{t'}(x) \leq |d_0| + O(\log(ni/\epsilon))$.

The "moreover" part can be easily seen as follows. Given a target string $x$ and a description $d_0$, compute $\mathrm{Dec}_{n,\epsilon}(U^A(d_0))$. Let $i$ be the index of $x$ in the list $\mathrm{Dec}_{n,\epsilon}(U^A(d_0))$. Output a description $(M, d_0, n, \epsilon^{-1}, i)$. □

Now we combine Lemma 4.21 and the list-decodable error-correcting code.

**Lemma 4.25.** *There exist some polynomial* poly *and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A string* $x \in \{0,1\}^*$ *of length* $n \in \mathbb{N}$*, parameters* $m, d/\ell, \delta^{-1} \in \mathbb{N}$ *represented in unary, and oracle access to* $T \subseteq \{0,1\}^m$*.*

Promise: *Let* $\epsilon := \delta/2m$*, and* $2^\ell := |\mathrm{Enc}_{n,\epsilon}(x)|$*. We assume that* $T$ *is a statistical test for* $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$ *with advantage* $\delta$*. That is,*

$$\left| \Pr_{z \sim \{0,1\}^d} \left[ T(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)) = 1 \right] - \Pr_{w \sim \{0,1\}^m} \left[ T(w) = 1 \right] \right| \geq \delta.$$

Output: *A certificate for* $\mathrm{K}^T_t(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\delta))$ *for any* $t \geq \mathsf{poly}(n, m, d, 1/\delta)$*.*

*Proof.* Set $f := \mathrm{Enc}_{n,\epsilon}(x) \in \{0,1\}^{2^\ell}$. Then run the algorithm of Lemma 4.21 with inputs $f, m, d/\ell, \delta^{-1}$ and oracle access to $T$. The algorithm outputs a certificate for $\mathrm{K}^T_{t_0,\,\delta/2m}(\mathrm{Enc}_{n,\epsilon}(x)) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(md))$, where $t_0 = \mathsf{poly}(m, d, 2^\ell)$. By Corollary 4.24, we may efficiently convert this certificate to a certificate for $\mathrm{K}^T_t(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\epsilon))$, where $t \geq t_0 + \mathsf{poly}(n, 1/\epsilon)$. □

As a consequence of Lemma 4.25, for any $x \in \{0,1\}^*$ and parameters with $d \gg \ell^2$, we may obtain a certificate of length $\approx \exp(\ell^2/d) \cdot m + d \approx m + \ell^2 m/d + d$ given a statistical test for $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$. Setting $d := \ell\sqrt{m}$, we obtain a certificate of length $\approx m + O(\ell\sqrt{m})$. We now claim that $m$ may be set to $\approx \mathrm{K}_t(x)$, by showing that the output of the Nisan-Wigderson generator is not random in the sense of time-bounded Kolmogorov complexity.

**Lemma 4.26.** *There exists some polynomial* poly *satisfying the following: For any* $n, \epsilon^{-1}, m, d/\ell \in \mathbb{N}$*,* $z \in \{0,1\}^d$ *and* $x \in \{0,1\}^n$ *(where* $2^\ell$ *is the output length of* $\mathrm{Enc}_{n,\epsilon}$*), we have*

$$\mathrm{K}_{t'}(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)) \leq \mathrm{K}_t(x) + d + O(\log(nmd/\epsilon))$$

*for any* $t, t' \in \mathbb{N}$ *with* $t' \geq t + \mathsf{poly}(n, 1/\epsilon, m, d)$*.*

*Proof.* Roughly speaking, the output of the Nisan-Wigderson generator can be described by a description $d_0$ of $x$ (which is of length $\mathrm{K}_t(x)$), and a seed $z$ of length $d$. More precisely, the following algorithm describes the output of the NW generator: Inputs consist of parameters $n, \epsilon^{-1}, m, d/\ell \in \mathbb{N}$ represented in binary, a seed $z \in \{0,1\}^d$, and a string $d_0 \in \{0,1\}^*$. The algorithm operates as follows. Compute $x := U(d_0)$, $f := \mathrm{Enc}_{n,\epsilon}(x)$, and the design $\mathcal{S}_{m,\ell,d}$. Output $\mathrm{NW}^f_{m,d}(z)$.

It is easy to see that the running time of this algorithm is at most $t + \mathsf{poly}(n, 1/\epsilon, m, d) \leq t'$, where $t$ denotes the time it takes for $U(d_0)$ to output $x$. The length of the description is at most $|d_0| + |z| + O(\log(nmd/\epsilon)) \leq \mathrm{K}_t(x) + d + O(\log(nmd/\epsilon))$. □

We now assume that an oracle $T$ is a $\delta$-dense subset of $r$-random strings $R_t[r]$. By Lemma 4.26, $T$ is a distinguisher for $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$ if $\mathrm{K}_t(x)+d \lesssim r(m)$. Thus by Lemma 4.25 we may find a certificate for $\mathrm{K}^T_{t'}(x) \precsim \exp(\ell^2/d) \cdot r^{-1}(\mathrm{K}_t(x)+d)+d$. A formal statement follows.

**Theorem 4.27.** *Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function. There exist some polynomial* $\mathsf{poly}$ *and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A string $x \in \{0,1\}^*$ of length $n \in \mathbb{N}$, parameters $t, m, d/\ell, \delta^{-1} \in \mathbb{N}$ represented in unary, and oracle access to $T \subseteq \{0,1\}^m$.*

Promise: *Let $\epsilon := \delta/2m$, and $2^\ell := |\mathrm{Enc}_{n,\epsilon}(x)|$. Assume that $T$ is a $\delta$-dense subset of $R_{t_1}[r]$ for some $t_1 \geq t+\mathsf{poly}(n, m, d, 1/\delta)$, and that $\mathrm{K}_t(x)+d+O(\log(nmd/\delta)) < r(m)$.*

Output: *A certificate for $\mathrm{K}^T_{t_2}(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\delta))$ for any $t_2 \geq \mathsf{poly}(n, m, d, 1/\delta)$.*

*Proof.* By Lemma 4.26, we have

$$\mathrm{K}_{t_1}(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)) \leq \mathrm{K}_t(x) + d + O(\log(nmd/\delta)) < r(m)$$

for any $z \in \{0,1\}^d$ and any $t_1 \geq t + \mathsf{poly}(n, m, d, 1/\delta)$. Therefore, for any $z \in \{0,1\}^d$, the output $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z)$ is not $r$-random with respect to $\mathrm{K}_{t_1}$; in particular, $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}(z) \notin T$. On the other hand, $\Pr_{w\sim\{0,1\}^m}[w \in T] \geq \delta$ by the assumption. Thus $T$ distinguishes $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(x)}_{m,d}$ from the uniform distribution with advantage at least $\delta$. By running the algorithm of Lemma 4.25 with input $x, m, d/\ell, \delta^{-1}$, and oracle access to $T$, the algorithm outputs a certificate for

$$\mathrm{K}^T_{t_2}(x) \preceq \exp(\ell^2/d) \cdot m + d + O(\log(nmd/\delta))$$

with high probability, where $t_2 \geq \mathsf{poly}(n, m, d, 1/\delta)$. □

By Theorem 4.27, for $r(m) \approx m$, we can set $m \approx \mathrm{K}_t(x)+d$; thus, we can find a certificate of length $\approx \exp(\ell^2/d)\cdot(\mathrm{K}_t(x)+d)+d \approx \mathrm{K}_t(x)+\ell^2\mathrm{K}_t(x)/d+2d+\ell^2$. By setting $d := \ell\sqrt{\mathrm{K}_t(x)}$, we obtain a certificate of length $\approx \mathrm{K}_t(x)+O(\ell\sqrt{\mathrm{K}_t(x)})+\ell^2$. (Note here that we do not know a priori the best choice of $d$ as well as $\mathrm{K}_t(x)$; however we can try all choices of $d$.) In the next corollary, we observe that the same length can be achieved as long as $m - O(\sqrt{m}\log m) \leq r(m)$.

**Corollary 4.28.** *Let $\delta^{-1} \in \mathbb{N}$ be any constant. Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that $m - c\sqrt{m}\log m \le r(m)$, for some constant $c$, for all large $m \in \mathbb{N}$. There exist some polynomial* poly *and a randomized polynomial-time oracle machine satisfying the following specification.*

Inputs: *A string $x \in \{0,1\}^*$ of length $n \in \mathbb{N}$, a parameter $t \in \mathbb{N}$ represented in unary, and oracle access to $T \subseteq \{0,1\}^*$.*

Promise: *For all large $m \in \mathbb{N}$, we assume that $T^{=m}$ is a $\delta$-dense subset of $R_{t_1}[r]$ for some $t_1 \ge t + \mathsf{poly}(n)$.*

Output: *A certificate for $\mathrm{K}^T_{t_2}(x) \preceq \mathrm{K}_t(x) + O\big((\log n)\sqrt{\mathrm{K}_t(x)} + (\log n)^2\big)$ for any $t_2 \ge \mathsf{poly}(n)$.*

*Proof.* Without loss of generality, we may assume that $O(\log n) \le \mathrm{K}_t(x) \le n + O(1)$. Indeed, we may exhaustively search all the descriptions $d_0$ of length $O(\log n)$ and check if $U(d_0)$ outputs $x$ within time $t$; if such a description is found, we may output $d_0$ as a certificate for $\mathrm{K}_t(x) \preceq |d_0| = O(\log n)$. Moreover, when $\mathrm{K}_t(x) \ge n + O(1)$, then we may just output a trivial description for $x$ of length $n + O(1)$. In what follows, we assume $O(\log n) \le \mathrm{K}_t(x) \le n + O(1)$.

Here is the algorithm: For every $m \in \{1, \dots, n + O(1)\}$ and every $d/\ell \in \{1, \dots, n + O(1)\}$, run the algorithm of Theorem 4.27 on input $x, t, m, d/\ell, \delta^{-1}$ and with oracle access to $T^{=m}$. Output the shortest description found in this way. (If none is found, output a trivial description for $x$ of length $n + O(1)$.)

We claim that, on some specific choice of $(m, d/\ell)$, the algorithm of Theorem 4.27 outputs a short description. Let $\ell := \log |\mathrm{Enc}_{n,\delta/2m}(x)| = O(\log n)$. We analyze the two cases depending on whether $\ell \le \sqrt{\mathrm{K}_t(x)}$ or not. Consider the case when $\ell \le \sqrt{\mathrm{K}_t(x)}$. Let $d/\ell := \sqrt{\mathrm{K}_t(x)}$ and $m := \mathrm{K}_t(x) + d + O(\log n) + 4c\sqrt{\mathrm{K}_t(x)}\log \mathrm{K}_t(x)$. Then we have $d \le \mathrm{K}_t(x)$ and hence $m \le 4\mathrm{K}_t(x)$. Thus,

$$\begin{aligned} r(m) &\ge m - c\sqrt{m}\log m \\ &\ge m - 2c\sqrt{\mathrm{K}_t(x)}\log 4\mathrm{K}_t(x) \\ &> \mathrm{K}_t(x) + d + O(\log n), \end{aligned}$$

which means the hypothesis of Theorem 4.27 is satisfied; therefore, with high probability, the algorithm outputs a description $d_0$ for $x$ such that $|d_0| \le \exp(\ell^2/d) \cdot m + d + O(\log n)$ with high probability. Since $\ell^2/d \le 1$, the length of the description is

$$\begin{aligned} |d_0| &\le (1 + 2\ell^2/d) \cdot m + d + O(\log n) \\ &\le (1 + 2\ell^2/d) \cdot (\mathrm{K}_t(x) + d) + 3 \cdot (O(\log n) + 4c\sqrt{\mathrm{K}_t(x)}\log \mathrm{K}_t(x)) + d + O(\log n) \\ &\le \mathrm{K}_t(x) + O(\ell\sqrt{\mathrm{K}_t(x)} + \ell^2). \end{aligned}$$

Next, consider the case when $\ell > \sqrt{\mathrm{K}_t(x)}$. In this case, let $d/\ell := \ell$ and $m := 4d$. Then we have $r(m) \ge m - c\sqrt{m}\log m \ge 3d > \mathrm{K}_t(x) + d + O(\log n)$, which confirms the hypothesis of Theorem 4.27. Thus the algorithm outputs a description for $x$ of length $\exp(1) \cdot m + d + O(\log n) = O(\ell^2)$. □

### 4.3.3 Accepting a Dense Subset of Random Strings in Heuristica

Now we justify the hypothesis used in the previous subsection. We show that a dense $r$-random string can be accepted by some polynomial-time machine if $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{AvgP}$. For any oracle $T \subseteq \{0,1\}^*$ and any $t \in \mathbb{N}$, let $T_t$

denote $\{\, x \in \{0,1\}^* \mid (x, 1^t) \in T \,\}$. The main idea here is that since there are few $r$-nonrandom strings, an errorless heuristic algorithm must succeed on a dense subset of $r$-random strings.

**Lemma 4.29.** *Let $r\colon \mathbb{N} \to \mathbb{N}$ be any function such that $r(n) < n$ for all large $n \in \mathbb{N}$. If $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_\delta\mathsf{P}$ for $\delta(m) := 1/6m$, then there exists a language $T \in \mathsf{P}$ such that $T_t^{=n}$ is a $\frac{1}{3}$-dense subset of $R_t[r]$, for all large $n \in \mathbb{N}$ and every $t \in \mathbb{N}$.*

*Proof.* Let $M$ be the errorless heuristic deterministic polynomial-time algorithm for $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}})$. We define $T$ so that $T(x, 1^t) := 1$ if $M(x, 1^t) = 0$; otherwise $T(x, 1^t) := 0$, for every $x \in \{0,1\}^*$ and $t \in \mathbb{N}$. By this definition, it is obvious that $T \in \mathsf{P}$.

Fix any $t \in \mathbb{N}$. We claim that $T_t$ is a subset of $r$-random strings $R_t[r]$. Indeed, for any $x \in T_t$, we have $M(x, 1^t) = 0$. Since $M$ is an errorless heuristic algorithm, we obtain $\mathrm{K}_t(x) \geq r(|x|)$; thus $x \in R_t[r]$.

We now claim that $T_t^{=n}$ is dense, i.e., $\Pr_{x\sim\{0,1\}^n}[x \in T_t] \geq \frac{1}{3}$ for all large $n \in \mathbb{N}$. In the next claim, we prove that $M$ solves $\mathrm{MINKT}[r]$ on average even if $t$ is fixed.

**Claim 4.30.** *For all large $n \in \mathbb{N}$ and any $t \in \mathbb{N}$, we have*

$$\Pr_{x\sim\{0,1\}^n}\left[M(x, 1^t) \neq \mathrm{MINKT}[r](x, 1^t)\right] \leq \frac{1}{6}.$$

Indeed, for $m := n + t$, using the definition of errorless heuristic algorithms, we obtain

$$\begin{aligned}
\delta(m) &\geq \Pr_{(x,1^s)\sim\mathcal{D}_m^{\mathrm{KT}}}\left[M(x, 1^s) \neq \mathrm{MINKT}[r](x, 1^s)\right] \\
&\geq \Pr\left[|x| = n\right] \cdot \Pr\left[M(x, 1^s) \neq \mathrm{MINKT}[r](x, 1^s) \mid |x| = n\right] \\
&\geq \frac{1}{m} \cdot \Pr_{x\sim\{0,1\}^n}\left[M(x, 1^t) \neq \mathrm{MINKT}[r](x, 1^t)\right],
\end{aligned}$$

where in the last inequality we used the fact that, conditioned on the event $|x| = n$, the distribution $\mathcal{D}_m^{\mathrm{KT}}$ is identically distributed to the distribution $(x, 1^t)$ where $x \sim \{0,1\}^n$. Thus we have $\Pr_{x\sim\{0,1\}^n}[M(x, 1^t) \neq \mathrm{MINKT}[r](x, 1^t)] \leq m \cdot \delta(m) \leq \frac{1}{6}$. This completes a proof of Claim 4.30.

We claim that $M$ must output 0 on a large fraction of strings, which implies that $T$ is dense. Indeed, there are few $r$-nonrandom strings, so $M$ must succeed on a large fraction of random strings. More precisely, the number of $r$-nonrandom strings of length $n$ is at most $\sum_{i=0}^{r(n)-1} 2^i \leq 2^{r(n)}$; thus, the probability that $(x, 1^t) \in \mathrm{MINKT}[r]$ over the choice of $x \sim \{0,1\}^n$ is at most $2^{r(n)-n} \leq \frac{1}{2}$, for all large $n \in \mathbb{N}$ and every $t \in \mathbb{N}$. Therefore, we obtain

$$\begin{aligned}
&\Pr_{x\sim\{0,1\}^n}\left[x \in T_t\right] \\
&= \Pr_x\left[M(x, 1^t) = 0 \;\&\; \mathrm{MINKT}[r](x, 1^t) = 0\right] \\
&= \Pr_x\left[M(x, 1^t) = \mathrm{MINKT}[r](x, 1^t)\right] \;-\; \Pr_x\left[M(x, 1^t) = 1 \;\&\; \mathrm{MINKT}[r](x, 1^t) = 1\right] \\
&\geq \left(1 - \frac{1}{6}\right) - \frac{1}{2} = \frac{1}{3}.
\end{aligned}$$

□

We will supply $T_t$ to the algorithm of Corollary 4.28; then the algorithm will output some certificate under the oracle $T_t$. The next lemma enables us to convert the certificate to a certificate under the oracle $T$.

**Lemma 4.31.** *There exists some polynomial* $\mathsf{poly}$ *such that* $\mathrm{K}^T_{t_3}(x) \le \mathrm{K}^{T_{t_1}}_{t_2}(x) + O(\log\log t_1)$ *for any* $x \in \{0,1\}^*$*, any oracle* $T \subseteq \{0,1\}^*$ *and any* $t_1, t_2, t_3 \in \mathbb{N}$ *such that* $t_1$ *is a power of* 2 *and* $t_3 \ge \mathsf{poly}(t_1, t_2)$*. Moreover, given* $t_1 \in \mathbb{N}$ *and a certificate for* $\mathrm{K}^{T_{t_1}}_{t_2}(x) \preceq s$*, one can efficiently find a certificate for* $\mathrm{K}^T_{t_3}(x) \preceq s + O(\log\log t_1)$*.*

*Proof.* Consider the following algorithm $M$ with oracle access to $T$: Given input $d_0 \in \{0,1\}^*$ and $\log t_1 \in \mathbb{N}$, simulate and output $U^{T_{t_1}}(d_0)$. Here, the oracle $T_{t_1}$ is simulated as follows: Given query $q$ to $T_{t_1}$, convert it into a query $(q, 1^{t_1})$ to $T$.

If $d_0$ is a certificate for $\mathrm{K}^{T_{t_1}}_{t_2}(x) \preceq |d_0|$, then $M^T(d_0, \log t_1)$ outputs $x$ in time $\mathsf{poly}(t_1, t_2)$. Thus $(M, d_0, \log t_1)$ is a certificate for $\mathrm{K}_{t_3}(x) \preceq |d_0| + O(\log\log t_1)$. □

In order to obtain a *zero-error* randomized algorithm for GapMINKT, we prove that a Kolmogorov-random string can be used in order to derandomize randomized algorithms. This can be proved by using the Nisan-Wigderson generator or the Impagliazzo-Wigderson generator (cf. [NW94, IW97, KvM02]); however, for our purpose, there is a much simpler construction of a pseudorandom generator based on Lemma 4.25. (Our construction is similar to the Sudan-Trevisan-Vadhan pseudorandom generator [STV01].)

**Lemma 4.32.** *For any constant* $\gamma > 0$*, there exist polynomials* $p_t, p_n$ *and a constant* $c \in \mathbb{N}$ *satisfying the following: For all large* $m \in \mathbb{N}$*, let* $t := p_t(m)$*,* $n := p_n(m)$*, and* $w \in \{0,1\}^n$ *be a string such that* $\mathrm{K}_t(w) \ge n^\gamma$*. Then,* $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(w)}_{m,d}$ *is a pseudorandom generator secure against a circuit of size* $m$*, where* $\epsilon := 1/4m$ *and* $d := c\log(nm)$*; that is,*

$$\left| \Pr_{z\sim\{0,1\}^d}\left[T(\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(w)}_{m,d}(z)) = 1\right] - \Pr_{u\sim\{0,1\}^m}\left[T(u) = 1\right] \right| < \frac{1}{2},$$

*for any circuit* $T$ *of size* $m$*.*

*Proof.* Let $\ell := \log|\mathrm{Enc}_{n,\epsilon}(w)| = O(\log(nm))$. Let $c$ be large enough so that $\exp(\ell^2/d) \le (nm)^{\gamma/2}$ for all large $m \in \mathbb{N}$. By Lemma 4.25, if $\mathrm{NW}^{\mathrm{Enc}_{n,\epsilon}(w)}_{m,d}$ is not a pseudorandom generator secure against some circuit $T$ of size $m$, then $\mathrm{K}^T_t(w) \le \exp(\ell^2/d)\cdot m + d + O(\log(nm)) \le (nm)^{\gamma/2}\cdot m + O(\log m)$ for $t := \mathsf{poly}(n, m, d)$. Since the circuit $T$ can be described by a string of length $O(m\log m)$, we obtain $\mathrm{K}_t(w) \le (nm)^{\gamma/2}\cdot m + O(m\log m)$. Thus we have $\mathrm{K}_t(w) < n^\gamma$ for some sufficiently large polynomials $n := p_n(m)$ and $t := p_t(m) \ge \mathsf{poly}(n, m, d)$, which is a contradiction. □

We arrive at the following search to average-case reduction.

**Theorem 4.33** (Restatement of Theorem 4.9)**.** *Let* $r\colon \mathbb{N} \to \mathbb{N}$ *be any function such that for some constant* $c > 0$*, for all large* $n \in \mathbb{N}$*,* $n - c\sqrt{n}\log n \le r(n) < n$*. Assume that* $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{1/6m}\mathsf{P}$*. Then, for some function* $\sigma(n,s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ *and some polynomial* $\tau$*, there exists a zero-error randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$*.*

*Proof.* We first present a randomized algorithm that may err. By Lemma 4.29, there exists a language $T$ in $\mathsf{P}$ such that $T_t^{=n}$ is a $\frac{1}{3}$-dense subset of $R_t[r]$ for all large $n \in \mathbb{N}$ and every $t \in \mathbb{N}$. Applying Corollary 4.28 to $T_{t_1}$ and $\delta^{-1} = 3$, we obtain a randomized polynomial-time oracle machine that, on input $x$ of length $n \in \mathbb{N}$, $1^t$, and with oracle access to $T_{t_1}$, outputs a certificate $d_0$ for $\mathrm{K}_{t_2}^{T_{t_1}}(x) \preceq \sigma(n, \mathrm{K}_t(x))$ with high probability, for $t_1 \geq t + \mathsf{poly}(n)$ and $t_2 \geq \mathsf{poly}(n)$. On input $(x, 1^t)$, we fix $t_1$ to the minimum integer such that $t_1$ is a power of 2 and $t_1 \geq t + \mathsf{poly}(n)$. By Lemma 4.31, we can efficiently transform the certificate $d_0$ into a certificate $d_1$ for $\mathrm{K}_{t_3}^{T}(x) \preceq \sigma(n, \mathrm{K}_t(x)) + O(\log\log t_1)$, where $t_3 \geq \mathsf{poly}(t_1, t_2)$. Since $T$ is solvable by a deterministic polynomial-time machine, by Lemma 4.16, we can efficiently transform the certificate $d_1$ into a certificate $d_2$ for $\mathrm{K}_{t_4}(x) \preceq \sigma(n, \mathrm{K}_t(x)) + O(\log\log t_1)$, where $t_4 \geq \mathsf{poly}(t_3)$. Thus we obtain a randomized polynomial-time algorithm that, on input $(x, 1^t)$, outputs a certificate $d_2$ for $\mathrm{K}_{t_4}(x) \preceq \sigma(|x|, \mathrm{K}_t(x)) + O(\log\log t_1)$ where $t_1 = \Theta(t + \mathsf{poly}(|x|))$ and $t_4 \geq \tau(|x|, t)$ for some polynomial $\tau$.

Note that there is an additive term $O(\log\log t_1)$; however, we may assume without loss of generality that $O(\log\log t_1) = O(\log n)$, which can be absorbed into $\sigma(n, \mathrm{K}_t(x))$. Indeed, otherwise we have $t_1 \geq 2^n$, and hence $t \geq \Omega(2^n)$. In such a case, we can exhaustively search all the description in time $\mathsf{poly}(t) = 2^{O(n)}$, and we can find the shortest description.

Now we make the randomized algorithm zero-error. Let $M$ denote the randomized algorithm solving the search version of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. Fix any input $(x, 1^t)$. Let $k, m, t'$ be some large polynomials in $|x|$ and $t$ that will be chosen later. Pick $w \sim \{0,1\}^k$ uniformly at random, and check if $w \in T_{t'}^{=k}$; note that $w$ is $r$-random since $T_{t'}^{=k}$ is a subset of $R_{t'}[r]$. Since $T_{t'}^{=n}$ is dense, we can find such an $r$-random string with high probability; if no $r$-random string is found, then output $\perp$ and halt (i.e., the zero-error algorithm fails). Using $w$ as a source of a hard function, we construct the secure pseudorandom generator $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{k,\epsilon}(w)}$ given in Lemma 4.32. Since the seed length of the pseudorandom generator is $O(\log(km))$, we can enumerate all the seeds $z$ in polynomial time; we use $\mathrm{NW}_{m,d}^{\mathrm{Enc}_{k,\epsilon}(w)}(z) \in \{0,1\}^m$ as the source of randomness of the randomized algorithm $M$. Output the shortest description that is found by exhaustively searching all the seeds.

To prove the correctness, we define some statistical test $T$. Fix any input $(x, 1^t)$, and let $C_M$ be a polynomial-size circuit that takes random bits $u$ and simulates the randomized algorithm $M$ on input $(x, 1^t)$ with random bits $u$. Let $s := \sigma(|x|, \mathrm{K}_t(x))$, $t'' := \tau(|x|, t)$, and let $V$ be a polynomial-size circuit that takes a description $d_0$ and accepts iff $d_0$ is a certificate for $\mathrm{K}_{t''}(x) \preceq s$. Define a statistical test $T$ as $T(u) := V(C_M(u))$. Let $m$ be large enough so that $|T| \leq m$ (for every choice of $s$); define $k := p_n(m)$ and $t' := p_t(m)$ where $p_n$ and $p_t$ are polynomials in Lemma 4.32. Since $M$ finds a certificate with high probability, we have $\Pr_{u \sim \{0,1\}^m}[T(u) = 1] \geq \frac{1}{2}$. Thus by Lemma 4.32, there exists some seed $z \in \{0,1\}^d$ such that $C_M(\mathrm{NW}_{m,d}^{\mathrm{Enc}_{k,\epsilon}(w)}(z))$ outputs a certificate for $\mathrm{K}_{t''}(x) \preceq s$, as desired. □

**Corollary 4.34.** *In the following list, we have* $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 4$. *Moreover, under the plausible assumption that* Promise-$\mathsf{ZPP}$ = Promise-$\mathsf{P}$, *we also have* $4 \Rightarrow 2$.

1. $\mathsf{DistNP} \subseteq \mathsf{AvgP}$.
2. $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{1/6m}\mathsf{P}$ *for some* $r \colon \mathbb{N} \to \mathbb{N}$ *such that* $n - O\big(\sqrt{n}\log n\big) \leq r(n) < n$ *for all large* $n \in \mathbb{N}$.

3. *There exists a zero-error randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$*, for some* $\sigma(n,s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ *and some polynomial* $\tau(n,t)$*.*

4. $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in$ Promise-$\mathsf{ZPP}$ *for some* $\sigma(n,s) = s + O\big((\log n)\sqrt{s} + (\log n)^2\big)$ *and some polynomial* $\tau(n,t)$*.*

*Proof.* ($1 \Rightarrow 2$) For $r(n) := n-1$, we have $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{DistNP} \subseteq \mathsf{AvgP} \subseteq \mathsf{Avg}_{1/6m}\mathsf{P}$.

($2 \Rightarrow 3$) This is exactly equivalent to Theorem 4.9.

($3 \Rightarrow 4$) This follows from Fact 4.15.

($4 \Rightarrow 2$ if Promise-$\mathsf{ZPP}$ = Promise-$\mathsf{P}$) Under the derandomization assumption, we have $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT} \in$ Promise-$\mathsf{ZPP}$ = Promise-$\mathsf{P}$. Let $c$ be a constant such that $\sigma(n,s) \le s + c \cdot \big((\log n)\sqrt{s} + (\log n)^2\big)$ for all large $n, s \in \mathbb{N}$. We claim that $(\mathrm{MINKT}[r], \mathcal{D}^{\mathrm{KT}}) \in \mathsf{Avg}_{\delta}\mathsf{P}$ for $r(n) := n - 2c(\log n)\sqrt{n}$ and $\delta(m) := 1/6m$.

By the assumption, there exists a deterministic polynomial-time algorithm $M$ that distinguishes Yes and No instances of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. Using the algorithm, we define an errorless heuristic algorithm $A$ solving $\mathrm{MINKT}[r]$ as follows: On input $(x, 1^t)$, if the input is short, i.e., $|x| = O(1)$, then check if $(x, 1^t) \in \mathrm{MINKT}[r]$ by an exhaustive search, and output the answer. Otherwise, set $s := r(|x|)$ and output 0 if $M(x, 1^t, 1^s)$ rejects; otherwise, output $\perp$.

We claim that $A$ is errorless. Since $A$ does not output 1 on inputs of large length, it suffices to claim that $M(x, 1^t, 1^s)$ accepts for any $(x, 1^t) \in \mathrm{MINKT}[r]$. Since $\mathrm{K}_t(x) < r(|x|) = s$, $(x, 1^t, 1^s)$ is a Yes instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$; thus $M(x, 1^t, 1^s)$ accepts.

We claim that $A$ succeeds on a large fraction of inputs. Fix any large $n \in \mathbb{N}$ and any $t \in \mathbb{N}$. For any $x \in \{0,1\}^n$, $A(x, 1^t)$ outputs $\perp$ only if $(x, 1^t, 1^{r(n)})$ is *not* a No instance of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$. Hence,

$$\begin{aligned}\Pr_{x\sim\{0,1\}^n}\big[A(x,1^t) = \perp\big] &\le \Pr_{x\sim\{0,1\}^n}\big[\mathrm{K}_{\tau(n,t)}(x) \le \sigma(n, r(n))\big] \\ &\le 2^{-n+\sigma(n,r(n))+1} \le \frac{1}{6},\end{aligned}$$

where the last inequality holds for all large $n$. Thus, for every $m \in \mathbb{N}$, we obtain

$$\Pr_{(x,1^t)\sim\mathcal{D}_m^{\mathrm{KT}}}\big[A(x,1^t) = \perp\big] = \mathop{\mathbb{E}}_{n\sim[m]}\Big[\Pr_{x\sim\{0,1\}^n}\big[A(x,1^{m-n}) = \perp\big]\Big] \le \frac{1}{6}.$$

□

## 4.4 Worst-Case to Average-Case Reduction for MCSP

In this section, we establish a worst-case and average-case equivalence for approximating a minimum circuit size. We start with the definition of randomized errorless heuristic algorithms.

**Definition 4.35** (Randomized Errorless Heuristics)**.** *Let* $(L, \mathcal{D})$ *be a distributional problem and* $\delta\colon \mathbb{N} \to [0,1]$*. A randomized algorithm* $A$ *is said to be a* randomized errorless heuristic algorithm *with failure probability* $\delta$ *for* $(L, \mathcal{D})$ *if*

- $\Pr_A\big[A(x) \notin \{L(x), \perp\}\big] \le \frac{1}{8}$ *for every* $x \in \{0,1\}^*$*, and*
- $\Pr_{x\sim\mathcal{D}_m}\big[\Pr_A[A(x) = \perp] > \frac{1}{8}\big] \le \delta(m)$ *for every* $m \in \mathbb{N}$*.*

*An input $x$ such that $\Pr_A[A(x) = \perp] > \frac{1}{8}$ is called a* hard instance *for $A$. We say that $(L, \mathcal{D}) \in \mathsf{Avg}_\delta\mathsf{BPP}$ if $(L, \mathcal{D})$ admits a randomized polynomial-time errorless heuristic algorithm with failure probability $\delta$. Define* $\mathsf{AvgBPP} := \bigcap_{c\in\mathbb{N}} \mathsf{Avg}_{n^{-c}}\mathsf{BPP}$.

Note that there are two types of randomness in this definition. One is the randomness of the BPP algorithm $A$, and the other is the randomness of the input $x$. The error $\frac{1}{8}$ that comes from the former can be reduced by the standard technique of repetition, and hence the choice $\frac{1}{8}$ is simply for the convenience.

We observe that an errorless heuristic algorithm for MCSP[$s$] is essentially equivalent to BPP-natural properties useful against $\mathsf{SIZE}(s(n))$. The main idea is that since the number of YES instances of MCSP[$s$] is small, an errorless heuristic algorithm must succeed on a large fraction of NO instances of MCSP[$s$], which induces a natural property.

**Lemma 4.36.** *Let $s\colon \mathbb{N} \to \mathbb{N}$ be any function such that $s(n) = o(2^n/n)$ for $n \in \mathbb{N}$. Let $\gamma, \delta\colon \mathbb{N} \to [0, 1]$ be functions.*

1. *If there exists a BPP-natural property useful against $\mathsf{SIZE}(s(n))$ with largeness $\gamma$, then $(\mathrm{MCSP}[s], \mathcal{U}) \in \mathsf{Avg}_\delta\mathsf{BPP}$, where $\delta(2^n) := 1 - \gamma(n)$ for $n \in \mathbb{N}$.*
2. *If $(\mathrm{MCSP}[s], \mathcal{U}) \in \mathsf{Avg}_\delta\mathsf{BPP}$, then there exists a BPP-natural property useful against $\mathsf{SIZE}(s(n))$ with largeness $\gamma$ where $\gamma(n) = 1 - \delta(2^n) - 2^{-2^{n-1}}$ for $n \in \mathbb{N}$.*

*Proof. First part:* Let $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ be a BPP-natural property, and $M$ be a BPP algorithm solving $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ (with error $\leq \frac{1}{8}$). Define a randomized algorithm $A$ as follows: On input $f$, run $M$ on input $f$ and reject if $M$ accepts, and output $\perp$ otherwise. We claim that $A$ is a randomized errorless heuristic algorithm for $(\mathrm{MCSP}[s], \mathcal{U})$.

We first claim that the fraction of hard instances for $A$ is small. Indeed, $f$ is a hard instance for $A$ only if $\Pr_A[A(f) = \perp] = \Pr_M[M(f) = 0] > \frac{1}{8}$, which implies that $f \notin \Pi_{\mathrm{YES}}$. Thus the fraction of hard instances $f \in \{0,1\}^{2^n}$ is at most $1 - \gamma(n)$.

Next, we claim that, for every input $f$, $A$ outputs a wrong answer for MCSP[$s$] with probability at most $\frac{1}{8}$. Since $A$ never accepts, this happens only if $M$ accepts and $f \in \mathrm{MCSP}[s]$, which implies that $f \in \Pi_{\mathrm{NO}}$ and hence $\Pr_A[A(f) = 0] = \Pr_M[M(f) = 1] \leq \frac{1}{8}$.

*Second part:* Given a randomized errorless heuristic algorithm $A$ for $(\mathrm{MCSP}[s], \mathcal{U})$, define a randomized algorithm $M$ so that $M(f) := 0$ if $A(f) = 1$ or $A(f) = \perp$; otherwise $M(f) := 1$. We claim that $M$ accepts some natural property $(\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ with error $\leq \frac{1}{4}$.

Since $A$ is errorless, for any $f \in \mathrm{MCSP}[s]$, we have $\Pr_M[M(f) = 1] = \Pr_A[A(f) = 0] \leq \frac{1}{8}$; thus $M$ satisfies the usefulness. To see the largeness, consider any instance $f$ that is not a hard instance for $A$. We claim that $\Pr_A[A(f) = \mathrm{MCSP}[s](f)] \geq \frac{3}{4}$: This is because $A$ outputs $\perp$ with probability at most $\frac{1}{8}$ since $f$ is not hard for $A$, and moreover $A$ outputs a wrong answer with probability at most $\frac{1}{8}$. Therefore, $M$ accepts $f$ with probability at least $\frac{3}{4}$ if $f$ is not a hard instance for $A$ and $f \notin \mathrm{MCSP}[s]$. The fraction of such instances $f \in \{0,1\}^{2^n}$ is at least $1 - \delta(2^n) - s(n)^{O(s(n))} \cdot 2^{-2^n} \geq \gamma(n)$, for all large $n \in \mathbb{N}$. □

We now state the main result of this section.

**Theorem 4.37.** *The following are equivalent.*

1. $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in$ Promise-$\mathsf{BPP}$ *for some* $\epsilon > 0$.
2. $(\mathrm{MCSP}[2^{\epsilon n}], \mathcal{U}) \in \mathsf{AvgBPP}$ *for some* $\epsilon > 0$.
3. $(\mathrm{MCSP}[2^{\epsilon n}], \mathcal{U}) \in \mathsf{Avg}_\delta\mathsf{BPP}$ *for some constants* $\epsilon, \delta \in (0,1)$.
4. *There exists a* $\mathsf{BPP}$*-natural property useful against* $\mathsf{SIZE}(2^{\epsilon n})$ *with largeness* $\gamma$*, for some* $\epsilon \in (0,1)$ *and* $\gamma(n) := 1 - 2^{-2^{n-1}}$.
5. *There exists a* $\mathsf{BPP}$*-natural property useful against* $\mathsf{SIZE}(2^{\epsilon n})$ *with largeness* $\gamma$*, for some constants* $\epsilon, \gamma \in (0,1)$.
6. *There exists a randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$*, for some* $\epsilon > 0$.

*Proof.* ($5 \Leftrightarrow 3$ and $4 \Rightarrow 2$) This follows from Lemma 4.36.

($2 \Rightarrow 3$) Obvious.

($6 \Rightarrow 1$) This follows from Fact 2.11.

($1 \Rightarrow 4$) Let $A$ be a randomized polynomial-time algorithm solving $\mathrm{Gap}_\epsilon\mathrm{MCSP}$. Define $A'$ as the following algorithm: On input $f\colon \{0,1\}^n \to \{0,1\}$, run $A$ on input $(f,s)$ for $s := 2^{\epsilon n/2}$, and accept iff $A$ rejects. We claim that $A'$ accepts some natural property useful against $\mathsf{SIZE}(2^{\epsilon n/2})$. Indeed, if $\mathsf{size}(f) \leq 2^{\epsilon n/2}$, then $(f,s)$ is a YES instance of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, and thus $A'(f)$ rejects with high probability. Hence $A'$ satisfies the usefulness. On the other hand, $A'$ accepts any NO instance $(f,s)$ of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, that is, any $(f,s)$ such that $\mathsf{size}(f) > 2^{(1-\epsilon)n} \cdot s = 2^{(1-\epsilon/2)n}$. Since the fraction of functions $f$ such that $\mathsf{size}(f) \leq 2^{(1-\epsilon/2)n}$ is at most $2^{O(n2^{(1-\epsilon/2)n}) - 2^n} \leq 2^{-2^n/2}$, $A'$ satisfies the largeness of density $1 - 2^{-2^n/2}$.

($5 \Rightarrow 6$) This is the main technical part, which can be proved by using a generic reduction from learning to natural properties [CIKK16]. We prove this in the next Theorem 4.38. □

**Theorem 4.38.** *If there exists a* $\mathsf{BPP}$*-natural property useful against* $\mathsf{SIZE}(2^{\epsilon_0 n})$ *with largeness* $\delta_0$ *for some constants* $\epsilon_0, \delta_0 \in (0,1)$*, then there exists a randomized polynomial-time algorithm solving the search version of* $\mathrm{Gap}_{\epsilon_1}\mathrm{MCSP}$ *for some* $\epsilon_1 > 0$.

For functions $f, g\colon \{0,1\}^n \to \{0,1\}$ and $\epsilon \in [0,1]$, we say that $f$ is $\epsilon$*-close* to $g$ if $\mathrm{dist}(f,g) \leq \epsilon$. The following is the main result of [CIKK16], which established a generic reduction from learning an $\epsilon$-close function to natural properties.

**Lemma 4.39** (Carmosino, Impagliazzo, Kabanets, and Kolokolova [CIKK16])**.** *For every* $\ell \leq n \in \mathbb{N}, \epsilon > 0$*, there exists a "black-box generator"* $G_{\ell,n,\epsilon}$ *satisfying the following.*

- $G_{\ell,n,\epsilon}$ *maps a function* $f\colon \{0,1\}^n \to \{0,1\}$ *to a function* $G^f_{\ell,n,\epsilon}\colon \{0,1\}^m \to \{0,1\}^{2^\ell}$ *for some* $m \in \mathbb{N}$*, and*
- $\mathsf{size}(G^f_{\ell,n,\epsilon}(z)) \leq \mathsf{poly}(n, 1/\epsilon, \mathsf{size}(f))$ *for all* $z \in \{0,1\}^m$*, where we regard* $G^f_{\ell,n,\epsilon}(z)$ *as a function on* $\ell$*-bit inputs.*

*Moreover, there exists a randomized polynomial-time oracle machine (a "reconstruction algorithm") satisfying the following specification.*

Inputs: *Oracle access to a function* $f\colon \{0,1\}^n \to \{0,1\}$, *parameters* $n, \epsilon^{-1}, 2^\ell \in \mathbb{N}$ *represented in unary, and a circuit* $D$ *on* $2^\ell$*-bit inputs.*

Promise: *We assume that* $D$ *is a statistical test for* $G^f_{\ell,n,\epsilon}$ *with advantage* $\delta_0$*. That is,*

$$\left| \Pr_{z\sim\{0,1\}^m}\Big[D(G^f_{\ell,n,\epsilon}(z))=1\Big] - \Pr_{w\sim\{0,1\}^{2^\ell}}\Big[D(w)=1\Big] \right| \geq \delta_0,$$

*for some universal constant* $\delta_0 > 0$*.*

Output: *A circuit* $C$ *that is* $\epsilon$*-close to* $f$*. (In particular, the size of* $C$ *is at most* $\mathsf{poly}(n, \epsilon^{-1}, 2^\ell, |D|)$*).*

*Proof of Theorem 4.38.* Suppose that the truth table of $f\colon \{0,1\}^n \to \{0,1\}$ is given as input. Let $u(\ell) := 2^{\epsilon_0 \ell}$ denote the usefulness parameter, and let $s := \mathsf{size}(f)$.

First, note that any circuit $C$ that is $\epsilon$-close to $f$ can be converted into a circuit $C'$ computing $f$ *exactly* such that $|C'| \leq |C| + \epsilon \cdot 2^n \cdot n + O(1)$. Indeed, since there are at most $\epsilon 2^n$ inputs on which $f$ and $C$ disagree, we can define a DNF formula $\varphi$ with $\epsilon 2^n$ terms such that $\varphi$ outputs 1 iff $f$ and $C$ disagree; then we may define $C'(x) := C(x) \oplus \varphi(x)$ so that $C'(x) = f(x)$ for every $x \in \{0,1\}^n$. Therefore, the output of the reconstruction algorithm of Lemma 4.39 can be converted to a circuit $C'$ computing $f$ exactly such that $|C'| \leq \mathsf{poly}(n, 1/\epsilon, 2^\ell, |D|) + \epsilon \cdot 2^n \cdot n$.

We now construct a statistical test for $G^f_{\ell,n,\epsilon}$ using a $\mathsf{BPP}$-natural property and Adleman's trick [Adl78] (for proving $\mathsf{BPP} \subseteq \mathsf{P/poly}$). Let $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$ be a $\mathsf{BPP}$-natural property and $M$ be a $\mathsf{BPP}$ algorithm solving $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$. Fix any $\ell \in \mathbb{N}$. By a standard error reduction for $\mathsf{BPP}$, we may assume without loss of generality that the error probability of $M$ is at most $2^{-2L}$ on any inputs of length $L := 2^\ell$. Pick a string $r$ of length $\mathsf{poly}(\ell)$ uniformly at random, and hardwire $r$ into $M$ as the source of internal randomness. Then, by the union bound, with probability at least $1 - 2^{-L}$ over the choice of $r$, $M(\cdot; r)$ computes some natural property on input length $L$, i.e., $M(w; r) = 1$ iff $w \in \Pi_{\text{YES}}$, for every $w \in (\Pi_{\text{YES}} \cup \Pi_{\text{NO}})^{=L}$. By a standard translation from a machine to a circuit, we convert $M(\cdot; r)$ to a circuit $D_\ell$ of size $\mathsf{poly}(\ell)$.

We claim that $D_\ell$ is a statistical test for $G^f_{\ell,n,\epsilon}$ if $\mathsf{size}(G^f_{\ell,n,\epsilon}(z)) \leq u(\ell)$ for every $z$. Indeed, by the usefulness of natural properties, we have $G^f_{\ell,n,\epsilon}(z) \in \Pi_{\text{NO}}$; thus $D_\ell(G^f_{\ell,n,\epsilon}(z)) = 0$. On the other hand, by the largeness of natural properties, we have $|(\Pi_{\text{YES}})^{=L}| \geq \delta_0 2^L$, and thus $\Pr_{w\sim\{0,1\}^L}[D_\ell(w) = 1] \geq \delta_0$. Thus $D_\ell$ is a statistical test for $G^f_{\ell,n,\epsilon}$.

Here is an algorithm solving the search version of $\mathrm{Gap}_{\epsilon_1}\mathrm{MCSP}$. For every $\ell \in [n]$ and every $\epsilon^{-1} \in [2^n]$, run the reconstruction algorithm of Lemma 4.39 with inputs $f, n, \ell, 2^\ell$, and $D_\ell$, and obtain a circuit $C$ approximating $f$. Convert $C$ to $C'$ computing $f$ exactly as explained above. Output the minimum circuit $C'$ computing $f$ found in this way.

It remains to claim that, for some choice of $\ell, \epsilon$, the reconstruction algorithm outputs a small circuit. Let $c > 0$ be a constant such that $G^f_{\ell,n,\epsilon}(z) \leq (ns/\epsilon)^c$ and $|C'| \leq (n2^\ell/\epsilon)^c + \epsilon 2^n n$ for all large $n, \ell, \epsilon^{-1}$. In order for $D_\ell$ to be a statistical test for $G^f_{\ell,n,\epsilon}$, we need $(ns/\epsilon)^c \leq u(\ell) = 2^{\epsilon_0 \ell}$; thus we set $2^\ell := (ns/\epsilon)^{c/\epsilon_0}$. To make $|C'|$ small, we set $\epsilon := 2^{-\epsilon_1 n} s$ where $\epsilon_1 := (c + c^2/\epsilon_0 + 1)^{-1}$. Then we have

$|C'| \leq (n/\epsilon)^c (n2^{\epsilon_1 n})^{c^2/\epsilon_0} + 2^{(1-\epsilon_1)n} ns \leq n^{O(1)} 2^{(1-\epsilon_1)n} s \leq 2^{(1-\epsilon_1/2)n} s$ for all large $n \in \mathbb{N}$. □

## 4.5 Open Problems

We conclude this chapter by mentioning several open questions. One obvious question is whether the approximation error of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ can be improved.

**Open Question 4.40.** Can the approximation error of $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ in Theorem 4.9 be improved?

Another question is whether a similar small approximation error can be achieved when we assume $\mathsf{AvgZPP}$ algorithms (instead of $\mathsf{AvgP}$) for MINKT. A naive approach is to have a description that incorporates random bits of $\mathsf{AvgZPP}$ algorithms, but it spoils the quality of the approximation.

**Open Question 4.41.** What approximation error can be achieved for $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ assuming $\mathsf{DistNP} \subseteq \mathsf{AvgZPP}$?

Finally and most importantly, we crucially used the fact that the algorithm does not make any error. Is it possible to obtain a similar non-black-box worst-case to average-case reduction for $\mathsf{HeurP}$ (i.e., heuristic algorithms that may err)?

**Open Question 4.42.** Does $\mathsf{DistNP} \subseteq \mathsf{HeurP}$ imply that $\mathrm{Gap}_{\sigma,\tau}\mathrm{MINKT}$ or $\mathrm{Gap}_{\epsilon}\mathrm{MCSP}$ is easy?

# Chapter 5

# On Black-box Reductions to Dense Subsets of Random Strings

In Chapter 4, we presented non-black-box reductions from an approximation version of MDLPs to its average-case version. As mentioned before, there is a line of work devoted to simulating black-box reductions by $\mathsf{AM} \cap \mathsf{coAM}$. A natural question is whether a similar technique can be used to show MDLPs are in $\mathsf{NP} \cap \mathsf{coAM}$ (and in particular refute average-case conjectures explored in Chapter 3.)

Thus in this chapter we investigate black-box reductions to dense subsets of random strings. As a main result, we show how to simulate randomized non-adaptive black-box reductions to dense subsets of (exponential-time computable) random strings by $\mathsf{AM} \cap \mathsf{coAM}$. We also show an upper bound of $\mathsf{S}_2^{\mathsf{NP}}$ in the case of reductions to dense subsets of random strings defined with respect to resource unbounded Kolmogorov complexity. These results further strengthen the evidence that the reductions of Chapter 4 are inherently non-black-box.

## 5.1 Background: Limits of Black-Box Reductions

A line of work was devoted to understanding why it is difficult to find a problem in $\mathsf{NP}$ whose average-case hardness is based on the worst-case complexity of an $\mathsf{NP}$-complete problem. Given our poor understanding of unconditional lower bounds, the most prevailing proof technique in complexity theory for showing intractability of a problem is by means of reductions. Moreover, almost all reduction techniques are *black-box* in the sense that, given computational problems $A$ and $B$, a reduction $R$ solves $A$ given any oracle (i.e. a black-box algorithm) solving $B$. The technique of reductions led to the discovery of tons of $\mathsf{NP}$-complete problems computationally equivalent to each other — in the *worst-case* sense. On the other hand, it turned out that the power of black-box reductions is limited for the purpose of showing intractability of average-case problems based on worst-case problems.

Building on the work of Feigenbaum and Fortnow [FF93], Bogdanov and Trevisan [BT06b] showed that if a worst-case problem $L$ is reducible to some average-case problem in $\mathsf{NP}$ via a nonadaptive black-box randomized polynomial-time reduction, then $L$ must be in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. This in particular shows that the hardness of any average-case problem in $\mathsf{NP}$ cannot be based on the worst-case hardness of an $\mathsf{NP}$-complete problem via such a reduction technique (unless the polynomial-time hierarchy collapses [Yap83]). Akavia, Goldreich, Goldwasser and Moshkovitz [AGGM06, AGGM10] showed that, in the special case of a non-adaptive reduction to the task of inverting a one-way function, the upper bound of [BT06b] can be improved to $\mathsf{AM} \cap \mathsf{coAM}$, thereby removing the advice "/poly".

Bogdanov and Brzuska [BB15] showed that even a general (i.e. adaptive) reduction to the task of inverting a size-verifiable one-way function cannot be used for any problem outside $\mathsf{AM} \cap \mathsf{coAM}$. Applebaum, Barak, and Xiao [ABX08] studied black-box reductions to PAC learning, and observed that the technique of [AGGM06] can be applied to (some restricted type of) a black-box reduction to the task of inverting an auxiliary-input one-way function.

The main question addressed in this chapter is whether the technique used in Chapter 4 is inherently non-black-box or not. As reviewed above, there are several results and techniques developed in order to simulate black-box reductions by $\mathsf{AM} \cap \mathsf{coAM}$ algorithms. Why can't we combine these techniques with the (seemingly non-black-box) reductions of Chapter 4 in order to prove $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{coAM}$ and refute average-case complexity conjectures explored in Chapter 3 such as Rudich's conjecture and O'Donnell's conjecture? Note that refuting these conjectures would significantly change our common belief about average-case complexity and the power of nondeterministic algorithms. We emphasize that while the proof of Chapter 4 seems to yield only non-black-box reductions, it does not necessarily mean that there is no alternative proof that yields a black-box reduction; moreover, the techniques of simulating black-box reductions could be adapted for the proof of Chapter 4.

For this purpose, we aim at improving our understanding of the limits of black-box reductions. We summarize a landscape around average-case complexity in Figure 5.1.

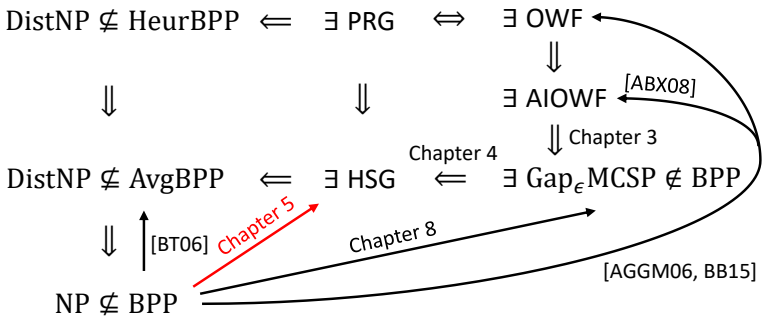


Figure 5.1: Average-case complexity and limits of black-box reductions. "$A \to B$" means that there is no black-box reduction technique showing "$A \Rightarrow B$" under reasonable complexity theoretic assumptions. The security of all cryptographic primitives is with respect to an almost-everywhere polynomial-time randomized adversary.

As indicated in Figure 5.1, we note that the reductions of Chapter 4 can be seen as a reduction to a problem of avoiding a hitting set generator. Indeed, $\mathrm{Gap}_\epsilon\mathrm{MCSP} \notin \mathsf{BPP}$ implies the nonexistence of natural properties, which yields a hitting set generator $G^{\mathsf{int}} = \{G^{\mathsf{int}}_{2^n} \colon \{0,1\}^{\widetilde{O}(2^{\epsilon' n})} \to \{0,1\}^{2^n}\}_{n \in \mathbb{N}}$ (i.e., the circuit interpreter defined in Definition 2.18) for some constant $\epsilon' > 0$.

We thus continue the study of the limits of black-box reductions to a distinguisher for a hitting set generator, initiated by Gutfreund and Vadhan [GV08]. Motivated by questions about whether derandomization is possible under uniform assumptions (cf. [IW01, TV07]), they investigated what can be reduced to any oracle avoiding a hitting set generator in a black-box way. They showed that any polynomial-time randomized nonadaptive black-box reductions to any oracle avoiding an exponential-time computable hitting set generator $G$ can be simulated in $\mathsf{BPP}^{\mathsf{NP}}$, which is a trivial upper bound when $G$ is polynomial-time

computable.

## 5.2 Our Results

We significantly improve this upper bound to $\mathsf{AM} \cap \mathsf{coAM}$, thereby putting the study of hitting set generators into the landscape of black-box reductions within $\mathsf{NP}$ (cf. Figure 5.1). We also show a uniform upper bound of $\mathsf{S}_2^{\mathsf{NP}}$ even if a hitting set generator $G$ is not computable.

**Theorem 5.1.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$ be any (not necessarily computable) hitting set generator such that $s(\ell) \leq (1 - \Omega(1))\ell$ for all large $\ell \in \mathbb{N}$. Then,*

$$\bigcap_R \mathsf{BPP}_{\|}^R \subseteq \mathsf{NP}/\mathsf{poly} \cap \mathsf{coNP}/\mathsf{poly} \cap \mathsf{S}_2^{\mathsf{NP}},$$

*where the intersection is taken over all oracles $R$ that $(1 - 1/\mathsf{poly}(\ell))$-avoid $G$. Moreover, if $G_\ell$ is computable in $2^{O(\ell)}$, then we also have*

$$\bigcap_R \mathsf{BPP}_{\|}^R \subseteq \mathsf{AM} \cap \mathsf{coAM}.$$

Compared to the line of work showing limits of black-box reductions within $\mathsf{NP}$, a surprising aspect of Theorem 5.1 is that it generalizes to any function $G$ that may not be computable. Indeed, almost all the previous results [FF93, BT06b, AGGM06, BB15, ABX08] crucially exploit the fact that a verifier can check the correctness of a certificate for an $\mathsf{NP}$ problem; thus a dishonest prover can cheat the verifier only for one direction, by not providing a certificate for a YES instance. In our situation, a verifier cannot compute $G$ and thus cannot prevent dishonest provers from cheating in this way. At a high level, our technical contributions are to overcome this difficulty by combining the ideas of Gutfreund and Vadhan [GV08] with the techniques developed in [FF93, BT06b].

We remark that Theorem 5.1 improves all the previous results mentioned before in some sense. Compared to [BT06b], our results show that the advice "/poly" is not required in order to simulate black-box reductions to any oracle avoiding an exponential-time computable hitting set generator. Compared to [AGGM06, ABX08], our results "conceptually" improve their results because the existence of one-way functions imply the existence of hitting set generators; on the other hand, since the implication goes through the *adaptive* reduction of [HILL99], technically speaking, our results are incomparable with their results. Similarly, our results conceptually improve the result of Chapter 8 (which concerns oracle-independent reductions), but these are technically incomparable, mainly because the implication goes through the non-black-box reduction of Chapter 4.

### 5.2.1 Why are the Reductions of Chapter 4 Non-black-box?

Based on Theorem 5.1, we now argue that the reductions of Chapter 4 are inherently non-black-box in a certain formal sense: The reason is that the idea of Chapter 4 can be applied to not only time-bounded Kolmogorov complexity but also any other types of Kolmogorov complexity, including resource-unbounded Kolmogorov complexity. Therefore, if this generalized reduction could be made black-box, then (as outlined below) by Theorem 5.1 we would obtain a finite algorithm $\mathsf{S}_2^{\mathsf{NP}}$ that approximates resource-unbounded Kolmogorov complexity, which is a contradiction.

To give one specific example, we briefly outline how the reductions of Chapter 4 can be generalized to the case of Levin's Kt-complexity [Lev84]: Recall that, for an efficient universal Turing machine $U$, the Kt-complexity of a string $x$ is defined as

$$\mathrm{Kt}(x) := \min\{|d| + \log t \mid U(d) \text{ outputs } x \text{ within } t \text{ steps } \}.$$

We define a hitting set generator $G$ as $G(d,t) := U(d)$ when $U(d)$ halts within $t$ steps, which is computable in exponential time. Note that $\mathrm{Im}(G)$ contains all strings with low Kt-complexity. Given an efficient algorithm $D$ that $\gamma$-avoids $G$, we can approximate $\mathrm{Kt}(x)$ by the following algorithm: Fix any input $x$. Take any list-decodable code Enc, and let $\mathrm{NW}^{\mathrm{Enc}(x)}(z)$ denote the Nisan-Wigderson generator [NW94] instantiated with $\mathrm{Enc}(x)$ as the truth table of a hard function, where $z$ is a seed of the generator. Then check whether the distinguishing probability $|\,\mathbb{E}_{z,w}[D(\mathrm{NW}^{\mathrm{Enc}(x)}(z)) - D(w)]|$ is large or small by sampling, whose outcome tells us whether $\mathrm{Kt}(x)$ is small or large, respectively. Indeed, if the distinguishing probability is large, then by using the security proof of the Nisan-Wigderson generator, we obtain a short description (with oracle access to $D$) for $x$. Conversely, if $\mathrm{Kt}(x)$ is small, then since $D$ $\gamma$-avoids $G$, the distinguishing probability is at least $\gamma$. Now, if we could make this analysis work for any oracle that $\gamma$-avoids $G$, then by Theorem 5.1 we would put a problem of approximating $\mathrm{Kt}(x)$ in $\mathsf{AM}$, which is not possible unless $\mathsf{EXP} = \mathsf{PH}$. (Note that the minimization problem of Kt is $\mathsf{EXP}$-complete under $\mathsf{NP}$ reductions [ABK+06b].)

### 5.2.2 Proof Overview

We outline our proof strategy for Theorem 5.1 below. Suppose that we have some reduction from $L$ to any oracle $R$ that breaks a hitting set generator $G$. Let $\mathcal{Q}$ denote the query distribution that a reduction makes.

As a warm-up, consider the case when the support $\mathsf{supp}(\mathcal{Q})$ of $\mathcal{Q}$ is small (*i.e.* $|\mathsf{supp}(\mathcal{Q}) \cap \{0,1\}^\ell| \ll 2^\ell$ for any length $\ell \in \mathbb{N}$). In this case, we can define an oracle $R_1$ so that $R_1 := \{0,1\}^* \setminus \mathsf{supp}(\mathcal{Q}) \setminus \mathrm{Im}(G)$; this breaks the hitting generator $G$ because $R_1$ is large whereas $R_1$ avoids the image of $G$ (which violates the definition of $G$ being a hitting set generator). Therefore, we can simulate the reduction by simply answering all the queries by saying "No"; hence such a reduction can be simulated in $\mathsf{BPP}$.

In general, we cannot hope that $\mathsf{supp}(\mathcal{Q})$ is small enough. To generalize the observation above, let us recall the notion of $\alpha$-heaviness [BT06b]: We say that a query $q$ is *$\alpha$-heavy* (with respect to $\mathcal{Q}$) if the query $q$ is $\alpha$ times more likely to be sampled under $\mathcal{Q}$ than the uniform distribution on $\{0,1\}^{|q|}$; that is, $\Pr_{w\sim\mathcal{Q}}[w = q] \geq \alpha 2^{-|q|}$. Now we define our new oracle $R_2 := \{0,1\}^* \setminus \{\, q \in \{0,1\}^* \mid q\colon \alpha\text{-heavy} \,\} \setminus \mathrm{Im}(G)$, which can be again shown to break $G$ because the fraction of $\alpha$-heavy queries is at most $1/\alpha$ $(\ll 1)$.

The problem now is that it is difficult to simulate the new oracle $R_2$; it appears that, given a query $q$, we need to test whether $q \stackrel{?}{\in} \mathrm{Im}(G)$, which is not possible in $\mathsf{AM} \cap \mathsf{coAM}$. However, it turns out that we do not need to test it, as we explain next: Observe that the size of $\mathrm{Im}(G)$ is very small; it is at most $2^{s(\ell)}$ $\left(\ll 2^\ell\right)$. Thus, the probability that a query $q$ is in $\mathrm{Im}(G)$ and $q$ is not $\alpha$-heavy (*i.e.* $q$ is rarely queried) is at most $\alpha \cdot 2^{s(\ell)-\ell}$, where $\ell$ is the length of $q$. As a consequence, the reduction cannot "distinguish" the oracle $R_2$ and a new oracle $R_3 := \{0,1\}^* \setminus \{\, q \in \{0,1\}^* \mid q\colon \alpha\text{-heavy} \,\}$; hence we can simulate the reduction if, given a query $q$, we are able to decide whether $q \stackrel{?}{\in} R_3$ in $\mathsf{AM} \cap \mathsf{coAM}$.

This task, however, still appears to be difficult for $\mathsf{AM} \cap \mathsf{coAM}$; indeed, at this point, Gutfreund and Vadhan [GV08] used the fact that the approximate counting is possible in $\mathsf{BPP}^{\mathsf{NP}}$, and thereby simulated the oracle $R_3$ by $\mathsf{BPP}^{\mathsf{NP}}$.

Our main technical contribution is to develop a way of simulating the reduction to $R_3$. First, note that the lower bound protocol of Goldwasser and Sipser [GS86] enables us to give an $\mathsf{AM}$ certificate for $\alpha$-heaviness; we can check, given a query $q$, whether $q$ is $\alpha(1+\epsilon)$-heavy or $\alpha$-light for any small error parameter $\epsilon > 0$. Thus, we have an $\mathsf{AM}$ protocol for $\{0,1\}^* \setminus R_3$ for every query $q$ (except for $\alpha(1 \pm \epsilon)$-heavy and light queries).

If, in addition, we had an $\mathsf{AM}$ protocol for $R_3$, then we would be done; unfortunately, it does not seem possible in general. The upper bound protocol of Fortnow [For89] does a similar task, but the protocol can be applied only for a limited purpose: we need to keep the randomness used to generate a query $q \sim \mathcal{Q}$ from being revealed to the prover. When the number of queries of the reduction is limited to 1, we may use the upper bound protocol in order to give an $\mathsf{AM}$ certificate for $R_3$; on the other hand, if the reduction makes two queries $(q_1, q_2) \sim \mathcal{Q}$, we cannot simultaneously provide $\mathsf{AM}$ certificates of the upper bound protocol for *both* of $q_1$ and $q_2$, because the fact that $q_1$ and $q_2$ are sampled *together* may reveal some information about the private randomness. To summarize, the upper bound protocol works only for the *marginal* distribution of each query, but does not work for the *joint* distribution of several queries.

That is, what we can obtain by using the upper bound protocol is information about *each* query. For example, the heavy-sample protocol of Bogdanov and Trevisan [BT06b] (which combines the lower and upper bound protocol and sampling) estimates, in $\mathsf{AM} \cap \mathsf{coAM}$, the probability that a query $q$ sampled from $\mathcal{Q}$ is $\alpha$-heavy.

Our idea is to overcome the difficulty above by generalizing the Feigenbaum-Fortnow protocol [FF93]. Feigenbaum and Fortnow developed an $\mathsf{AM} \cap \mathsf{coAM}$ protocol that simulates a nonadaptive reduction to an $\mathsf{NP}$ oracle $R$, given as advice the probability that a query is a positive instance of $R$. We generalize the protocol in the case when the oracle $\{0,1\}^* \setminus R_3$ is solvable by $\mathsf{AM}$ on average (which can be done by the lower bound protocol [GS86]), and given as advice the probability that a query $q$ is in $\{0,1\}^* \setminus R_3$ (which can be estimated by the heavy-sample protocol [BT06b]):

**Theorem 5.2** (Generalized Feigenbaum-Fortnow Protocol; informal)**.** *Suppose that $M$ is a randomized polynomial-time nonadaptive reduction to oracle $R$ whose queries are distributed according to $\mathcal{Q}$, and that $R$ is solvable by $\mathsf{AM}$ on average (that is, there exists an $\mathsf{AM}$ protocol $\Pi_R$ such that, with probability $1 - 1/\mathsf{poly}(n)$ over the choice of $q \sim \mathcal{Q}$, the protocol $\Pi_R$ computes $R$ on input $q$). Then, there exists an $\mathsf{AM} \cap \mathsf{coAM}$ protocol $\Pi_M$ such that, given a probability $p^* \approx \Pr_{q \sim \mathcal{Q}}[q \in R]$ as advice, the protocol $\Pi_M$ simulates the reduction $M$ with probability at least $1 - 1/\mathsf{poly}(n)$.*

**Organization.** The rest of this chapter is organized as follows. After reviewing necessary background in Section 5.3, we show that a reduction only with short queries can be simulated by $\mathsf{S}_2^{\mathrm{p}}$ in Section 5.4. We present the generalized Feigenbaum-Fortnow protocol in Section 5.5; then the proof of Theorem 5.1 is completed in Section 5.6 by showing that long queries can be simulated by $\mathsf{AM} \cap \mathsf{coAM}$.

## 5.3 Preliminaries

### Interactive Proof Systems

$\mathsf{AM}$ is the class of languages $L$ that can be accepted by some polynomial-time two-round Arthur-Merlin protocol; that is, there exists a polynomial-time Turing machine $V$ (called an $\mathsf{AM}$ verifier) such that

- (Completeness) if $x \in L$ then $\Pr_r[V(x, y, r) = 1 \text{ for some } y] \geq \frac{2}{3}$, and
- (Soundness) if $x \notin L$ then $\Pr_r[V(x, y, r) = 1 \text{ for some } y] \leq \frac{1}{3}$.

For our purpose, it is convenient to use the following characterization of $\mathsf{AM} \cap \mathsf{coAM}$.

**Fact 5.3.** *Let $L \subseteq \{0,1\}^*$. Then $L \in \mathsf{AM} \cap \mathsf{coAM}$ if and only if there exists a randomized polynomial-time verifier $V$ of a private-coin constant-round interactive proof system such that, for any input $x \in \{0,1\}^*$,*

- *(Completeness) there exists a prover $P$ such that $V$ outputs $L(x)$ by communicating with $P$ with probability at least $\frac{2}{3}$, and*
- *(Soundness) for any prover $P$, $V$ outputs $L(x)$ or $\perp$ with $P$ with probability at least $\frac{2}{3}$.*

That is, the verifier outputs a correct answer $L(x)$ when interacting with an honest prover, and it does not output the wrong answer $1 - L(x)$ for any cheating prover, with high probability. The fact above follows from the transformation from public coin protocols to private coin protocols [GS86], and the $\mathsf{AM}$ hierarchy collapses [BM88].

$\mathsf{S}_2^{\mathrm{p}}$ (the second level of the symmetric alternation [Can96, RS98]) is the class of languages $L$ such that a polynomial-time verifier decides acceptance based on certificates given by two competitive provers. More formally, $L \in \mathsf{S}_2^{\mathrm{p}}$ if and only if there exists a polynomial-time verifier $V$ such that $\exists y, \forall z, L(x) = V(x, y, z)$ and $\exists z, \forall y, L(x) = V(x, y, z)$, for every input $x \in \{0,1\}^*$. We say that $L \in \mathsf{S}_2^{\mathsf{NP}}$ if the verifier $V$ is given oracle access to any $\mathsf{NP}$ oracles in the definition above.

### Circuits

We will use a circuit in order to make it easy to compose several protocols[1]. Usually, a circuit takes a string of some fixed length as input and outputs a string of some fixed length. For our purpose, it is convenient to extend this usual notion of circuit: By using some encoding, we regard circuits as taking a string of length *at most* $l$ for some $l \in \mathbb{N}$, and outputting a string (not necessarily of fixed length). We regard a circuit as computing a function from $\{0,1\}^*$ to $\{0,1\}^* \cup \{\text{“undefined”}\}$ such that the function outputs “undefined” on inputs of length $> l$.

### Nonadaptive Reductions

A polynomial-time *nonadaptive* reduction is a polynomial-time oracle Turing machine whose possible queries can be computed without access to an oracle in polynomial time. For simplicity, we assume without loss of generality that,

[1]The reader may simply regard a circuit as a Turing machine with an appropriate description to which one can embed some additional information.

for all inputs of the same length, the reduction makes the same number $m$ of queries (by adding dummy queries if necessary). For any oracle $R \subseteq \{0,1\}^*$, we denote by $\mathsf{BPP}_{\|}^R$ the class of languages from which there exists a randomized polynomial-time nonadaptive reduction. For a nonadaptive reduction $M$, we denote by $M^R(x)$ the output of the reduction given an oracle $R \subseteq \{0,1\}^*$ and input $x \in \{0,1\}^*$.

### Query Distribution

We can modify a randomized nonadaptive reduction $M$ so that the marginal distribution of each query of $M$ is identical; that is, for any query $q \in \{0,1\}^*$, the probability that $q$ is sampled as the $i$th query of $M$ is the same for all $i \in [m]$. To achieve this, we simply modify $M$ as follows: it generates a permutation $\pi\colon [m] \to [m]$ uniformly at random, runs $M(x)$ to make $m$ queries $q_1, \ldots, q_m$, asks $q_{\pi(i)}$ as the $i$th query to get an answer $a_i$ from an oracle, and resumes the computation of $M(x)$ to get the decision on $x$ by supplying $a_{\pi^{-1}(1)}, \ldots, a_{\pi^{-1}(m)}$ as oracle answers. It is then easy to see that in the new query machine the $i$th query distribution is identical for all $i \in [m]$. By the modification above, we can take a single query distribution $\mathcal{Q}_x$ such that each query of $M(x)$ is distributed according to $\mathcal{Q}_x$.

Let $\mathcal{Q}$ be a distribution over $\{0,1\}^*$. For a string $x \in \{0,1\}^*$, let $\mathcal{Q}(x)$ denote $\Pr_{q\sim\mathcal{Q}}[q = x]$. For any $\alpha > 0$, a string $q \in \{0,1\}^*$ of length $\ell \in \mathbb{N}$ is called *$\alpha$-heavy* (with respect to $\mathcal{Q}$) if $\mathcal{Q}(q) \geq \alpha 2^{-\ell}$; otherwise (i.e., $\mathcal{Q}(q) < \alpha 2^{-\ell}$), it is called *$\alpha$-light*.

We will use a standard concentration inequality:

**Lemma 5.4** (Hoeffding's inequality [Hoe63])**.** *For any independent random variables $X_1, \ldots, X_n \in [0,1]$ and any $t \geq 0$, we have* $\Pr\left[\left|\sum_{i=1}^n (X_i - \mathbb{E}[X_i])\right| \geq nt\right] \leq 2\exp(-2nt^2)$.

### Black-Box Reductions

There are two possible definitions of black-box reductions to avoiding hitting set generator. One is to require that there exists a *single* machine that works for *every* $\gamma$-avoiding oracle. This is the notion used in [GV08]:

**Definition 5.5** (Black-box reduction to $\gamma$-avoiding oracles [GV08])**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ be any description interpreter. Let $L \subseteq \{0,1\}^*$ be a language. A randomized nonadaptive oracle machine $M$ is called a* black-box reduction *from $L$ to any $\gamma$-avoiding oracle of $G$ if, for any $\gamma$-avoiding oracle $R$ for $G$ and any $x \in \{0,1\}^*$, we have*

$$\Pr\left[\, M^R(x) = L(x) \,\right] \geq \frac{2}{3}, \tag{5.1}$$

*where the probability is taken over the internal randomness of $M$.*

Another definition is that, for every $\gamma$-avoiding oracle $R$, there exists a nonadaptive reduction $M_R$ to $R$ as used in Theorem 5.1. That is, the order of the quantifiers is reversed; nonetheless, a diagonalization argument enables us to establish the equivalence:

**Proposition 5.6.** *Let $G$ be any description interpreter, $\gamma\colon \mathbb{N} \to [0,1)$ be any parameter, and $L \subseteq \{0,1\}^*$ be a language. The following are equivalent:*

1. $L \in \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPP}_{\|}^{R}$.

2. *There exists a randomized polynomial-time nonadaptive black-box reduction from $L$ to any $\gamma$-avoiding oracle of $G$.*

*Proof.* The direction from the second item to the first item is obvious. We prove below the contrapositive of the other direction.

Suppose that, for any randomized nonadaptive oracle machine $M$, there exists some $\gamma$-avoiding oracle $R_M$ of $G$ such that $\Pr_M\left[M^R(x) = L(x)\right] < \frac{2}{3}$ for some $x \in \{0,1\}^*$. We claim that there exists some *single* $\gamma$-avoiding oracle $R$ of $G$ such that $L \notin \mathsf{BPP}_{\|}^{R}$.

To this end, let $\{M_e\}_{e\in\mathbb{N}}$ be the set of all randomized nonadaptive oracle machines. We will construct some $\gamma$-avoiding oracle $R_e$ and input $x_e$ (and $\ell_e \in \mathbb{N}$) by induction on $e \in \mathbb{N}$, so that $M_e$ given oracle $R_{e+1}$ fails to compute $L$ on input $x_e$; then we will define $R := \bigcup_{e\in\mathbb{N}} R_e$. Let us start with $R_0 := \varnothing$ and $\ell_0 := 0$.

At stage $e \in \mathbb{N}$, we claim that there exists some $\gamma$-avoiding oracle $R'_{e+1} \subseteq \{0,1\}^*$ and some input $x_e \in \{0,1\}^*$ such that

- $\Pr\left[M_e^{R'_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3}$, and
- $q \in R_e$ if and only if $q \in R'_{e+1}$ for any string $q$ of length $< \ell_e$.

Indeed, for any oracle $Q$, let $Q' := \{\, q \in Q \mid |q| \geq \ell_e \,\} \cup \{\, q \in R_e \mid |q| < \ell_e \,\}$. Consider a randomized nonadaptive oracle machine $M'_e$ such that $M_e'^{Q}$ simulates $M_e^{Q'}$; that is, $M'_e$ is hardwired with the set $\{\, q \in R_e \mid |q| < \ell_e \,\}$, and simulates $M_e$ and answer any query $q$ of length $< \ell_e$ by using the hardwired information. By our assumption, there exists some $\gamma$-avoiding oracle $\hat{R}_{e+1}$ of $G$ such that $\Pr\left[M_e'^{\hat{R}_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3}$ for some $x_e \in \{0,1\}^*$; by the definition of $M'_e$, we obtain $\Pr\left[M_e^{R'_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3}$ for $R'_{e+1} := \{\, q \in \hat{R}_{e+1} \mid |q| \geq \ell_e \,\} \cup \{\, q \in R_e \mid |q| < \ell_e \,\}$, which is again $\gamma$-avoiding $G$. This completes the proof of the claim above. Now define $\ell_{e+1} \in \mathbb{N}$ as a large enough integer so that $\ell_{e+1} \geq \ell_e$ and the machine $M_e$ on input $x_e$ does not query any string of length $\geq \ell_{e+1}$, and define an oracle $R_{e+1} := \{\, q \in R'_{e+1} \mid |q| < \ell_{e+1} \,\}$, which completes the construction of stage $e \in \mathbb{N}$.

Define $R := \bigcup_{e\in\mathbb{N}} R_e$, which $\gamma$-avoids $G$ by the construction above. By the choice of $(\ell_e)_{e\in\mathbb{N}}$, we have

$$\Pr\left[M_e^R(x_e) = L(x_e)\right] = \Pr\left[M_e^{R_{e+1}}(x_e) = L(x_e)\right] < \frac{2}{3},$$

for every randomized nonadaptive oracle machine $M_e$. Thus $L \notin \mathsf{BPP}_{\|}^{R}$. □

## 5.4 Simulating Short Queries by Competitive Prover Systems

In this section, we show that a reduction that makes only short queries can be simulated by $\mathsf{S}_2^{\mathrm{p}}$.

**Theorem 5.7** ($\mathsf{S}_2^{\mathrm{p}}$ Simulation of Short Queries)**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^{\ell}\}_{\ell\in\mathbb{N}}$ be any description interpreter and $\gamma\colon \mathbb{N} \to [0,1)$ be a parameter such that $\gamma(\ell) \leq 1 - 2^{s(\ell)-\ell+1}$ for all large $\ell \in \mathbb{N}$. Suppose that there exists a randomized polynomial-time black-box reduction $M$ from a language $L$ to any $\gamma$-avoiding oracle for $G$ such that the length of any query of $M$ is at most $O(\log n)$ for every input of length $n$. Then $L \in \mathsf{S}_2^{\mathrm{p}}$.*

*Proof.* The idea is that two competitive provers send the image $\mathrm{Im}(G)$ of $G$ as a certificate. Given two possible images $I_0, I_1 \subseteq \{0,1\}^*$, $R := \{0,1\}^* \setminus I_0 \setminus I_1$ is an avoiding set for $G$. Moreover, since $|\mathrm{Im}(G)|$ is small, the set $R$ is dense enough. We then derandomize a $\mathsf{BPP}$ computation by using the power of $\mathsf{S}_2^{\mathrm{p}}$.[2] (Recall that $\mathsf{BPP} \subseteq \mathsf{S}_2^{\mathrm{p}}$ [Can96, RS98].) Details follow.

Let $c \log n$ be an upper bound on the length of queries that $M$ makes. Our $\mathsf{S}_2^{\mathrm{p}}$ algorithm is as follows: Fix any input $x$ of length $n$. We number the two competitive provers 0 and 1. The $i$th prover ($i \in \{0,1\}$) sends, for each $\ell \leq c \log n$, a subset $I_{i,\ell} \subseteq \{0,1\}^\ell$ of size at most $2^{s(\ell)}$; an honest prover sets $I_{i,\ell} := \mathrm{Im}(G_\ell)$. Define $I_i := \bigcup_{\ell \leq c \log n} I_{i,\ell}$. Note that such subsets can be encoded as a string of polynomial length. Each prover also sends a list of randomness $r_1^i, \cdots, r_t^i \in \{0,1\}^m$ to be used by the reduction $M$, where $t$ is a parameter chosen later, and $m$ is the length of a coin flip used by $M$. The verifier sets $R := \{0,1\}^* \setminus I_0 \setminus I_1$, and accept if and only if $\Pr_{j,k \sim [t]}[M^R(x; r_j^0 \oplus r_k^1) = 1] > \frac{1}{2}$, where $M^R(x;r)$ denotes the output of the reduction when its coin flip is $r$, and $\oplus$ denotes the bit-wise XOR. Note that the running time of the verifier is at most a polynomial in $n$ and $t$. Below we establish the correctness of this algorithm for some $t = \mathsf{poly}(n)$.

We focus on the case when the 0th prover is honest; thus $I_{0,\ell} := \mathrm{Im}(G_\ell)$ for each $\ell \leq c \log n$. Since $|I_{1,\ell}| \leq 2^{s(\ell)}$, the number of strings of length $\ell$ in $R(I_1) := \{0,1\}^* \setminus I_0 \setminus I_1$ is at least $2^\ell - |I_{0,\ell}| - |I_{1,\ell}| \geq 2^\ell \gamma(\ell)$ (here we write $R(I_1)$ instead of $R$ to emphasize that $R$ depends on $I_1$); thus $R(I_1)$ is a $\gamma$-avoiding oracle. By the definition of the reduction, for every $I_1$ we have $\Pr_{r \sim \{0,1\}^m}[M^{R(I_1)}(x;r) = L(x)] \geq \frac{2}{3}$.

We use the notion of cover introduced by Canetti [Can96]: A sequence $r_1, \cdots, r_t \in \{0,1\}^m$ is called a cover of a subset $A \subseteq \{0,1\}^m$ if for all $r \in \{0,1\}^m$, $\Pr_{j \sim [t]}[r_j \oplus r \in A] > \frac{1}{2}$. Define $A(I_1) := \{\, r \in \{0,1\}^m \mid M^{R(I_1)}(x;r) = L(x) \,\}$. We claim that by a probabilistic argument there exists a sequence $r_1, \cdots, r_t \in \{0,1\}^m$ that covers $A(I_1)$ for every $I_1$: Fix any $I_1$ and $r \in \{0,1\}^m$. Pick $r_1, \cdots, r_t \sim \{0,1\}^m$. For any $j \in [t]$, the probability that $r_j \oplus r \in A(I_1)$ is at least $\frac{2}{3}$. Thus by a concentration bound (Lemma 5.4), the probability that at most a $\frac{1}{2}$-fraction of $j \in [t]$ satisfies $r_j \oplus r \in A(I_1)$ is at most $\exp(-\Omega(t))$. By the union bound over all $r$, the probability that a sequence $r_1, \cdots r_t$ does not cover $A(I_1)$ is at most $2^m \cdot \exp(-\Omega(t))$. By the union bound over all $I_1$, the probability that there exists some $I_1$ such that $A(I_1)$ is not covered by $r_1, \cdots, r_t$ is at most $2^{n^c+m} \cdot \exp(-\Omega(t))$. Therefore, for $t := \Theta(n^c + m)$, there exists a sequence $r_1, \cdots, r_t$ that covers $A(I_1)$ for every $I_1$. The 0th honest prover sends this sequence $r_1, \cdots, r_t$ to the verifier as $r_1^0, \cdots, r_t^0$, in which case the verifier outputs $L(x)$ correctly because $\Pr_{j,k \sim [t]}[M^{R(I_1)}(x; r_j^0 \oplus r_k^1) = L(x)] > \frac{1}{2}$, for every $I_1$ and every $r_1^1, \cdots, r_t^1$. □

## 5.5 Generalized Feigenbaum-Fortnow Protocol

In this section, we present one of the main building blocks of our proof. Our protocol is inspired by the protocol of Feigenbaum and Fortnow [FF93] (and its description by Bogdanov and Trevisan [BT06b]) for simulating some type of randomized nonadaptive reduction $M$ to an $\mathsf{NP}$ problem $R$. Suppose that for a given input $x$, $M$ makes $m$ nonadaptive queries $q_1, \ldots, q_m$ under a certain distribution $\mathcal{Q}$. In the Feigenbaum-Fortnow protocol, a verifier asks a prover to give witnesses to all positive instances among them. The prover cannot give a witness to a negative instance (hence, it cannot cheat the verifier by saying "yes" to a

[2] Alternatively, we may use the result of Russell and Sundaram [RS98] showing that $\mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P} = \mathsf{S}_2^{\mathrm{p}}$ in a black-box way.

negative instance) while it may try to cheat the verifier by not giving a witness to some of the positive instances of $q_1, \ldots, q_m$. If, however, the verifier knows the proportion $p^*$ of positive instances among queries under the distribution $\mathcal{Q}$, then it may detect wrong negative answers from the prover if the number of positive answers is much smaller than $p^*m$. More specifically, the Feigenbaum-Fortnow protocol runs as follows. It first generates $K$ tuples of $m$ nonadaptive queries $\{(q_{k1}, \ldots, q_{km})\}_{1 \le k \le K}$ by running $M(x)$ independently $K$ times. By a concentration inequality, the number of positive instances among all $Km$ queries should be in the range of $m \cdot \left(p^*K \pm O(\sqrt{K})\right)$ with high probability; thus, if the prover gives "yes" answers (with witnesses) much smaller than $m \cdot \left(p^*K - O(\sqrt{K})\right)$, then the verifier stops the computation immediately, suspecting that the prover is not honest. On the other hand, if the number of positive answers to those queries is close to $p^*Km$, then the number of positive instances on which the prover can cheat is at most $O(m\sqrt{K})$, with high probability. We choose $K$ large enough so that $O(m\sqrt{K}) \ll K$; then the majority of $K$ tuples are answered correctly by the oracle, and we can use them to determine the result of $M^R(x)$ by taking the majority vote of the results of $M(x)$ computed by using prover's answers to each tuple of queries $(q_{k1}, \ldots, q_{km})$.[3]

We generalize the Feigenbaum-Fortnow protocol so that a new protocol is capable of dealing with a reduction to a *distributional* $\mathsf{AM}$ problem $R$; that is, we show that, given any nonadaptive reduction to some $\mathsf{AM}$ problem solvable on average and the proportion $p^*$ of positive instances as advice, one can simulate the reduction in $\mathsf{AM} \cap \mathsf{coAM}$. In our protocol, we use Adleman's trick (for proving $\mathsf{BPP} \subseteq \mathsf{P/poly}$ [Adl78]) to "derandomize" $\mathsf{AM}$ oracle so that we obtain a new $\mathsf{NP}$ oracle, and then run the original Feigenbaum-Fortnow protocol. The following is the specification of the generalized Feigenbaum-Fortnow protocol:

---

**Inputs.** A tuple $(C, V, \delta, p^*)$ such that:

- A randomized nonadaptive reduction $C$ is given as a probabilistic circuit such that each query of $C$ is identically distributed to some distribution $\mathcal{Q}$ over $\{0,1\}^*$, and the reduction always makes exactly $m$ queries. (We assume that an input to a reduction is hardwired into the circuit $C$; thus $C$ does not take any input other than random bits.)
- An $\mathsf{AM}$ verifier $V$ is given as a circuit.
- An error parameter $\delta \in \left(0, \frac{1}{2}\right)$ is given in unary, and a probability $p^* \in [0,1]$ is given in binary.

**Promise.** We assume that there exist some answer $a \in \{0,1\}$, some oracle $R \subseteq \{0,1\}^*$, and some error parameters $\epsilon_0, \epsilon_1, \epsilon_2 \in [0,1]$ satisfying the following:

- $\Pr_C[C^R = a] \ge 1 - \epsilon_0$. (That is, $a$ is supposed to be the answer of the reduction $C$ to the oracle $R$.)
- The advice $p^*$ satisfies $|p^* - \Pr_{q \sim \mathcal{Q}}[q \in R]| \le \epsilon_1$.

---

[3] We note that taking the majority is not necessary; instead, it suffices to pick $k \sim [K]$ and use the result.

- The distributional problem $(R, \mathcal{Q})$ is "solvable by AM on average": that is, define[4]

$$V_{\text{YES}} := \{\, q \in \{0,1\}^* \mid \Pr_r[V(q,y,r) = 1 \text{ for some } y] \geq 3/4 \,\}, \text{ and}$$
$$V_{\text{NO}} := \{\, q \in \{0,1\}^* \mid \Pr_r[V(q,y,r) = 0 \text{ for all } y] \geq 3/4 \,\};$$

then we assume that $\Pr_{q\sim\mathcal{Q}}[q \in V_{\text{YES}} \cup V_{\text{NO}}] \geq 1 - \epsilon_2$ and $V_{\text{YES}} \subseteq R \subseteq \{0,1\}^* \setminus V_{\text{NO}}$.

**Protocol.**

1. (Preprocess of Verifier: Adleman's trick) Let $s$ be sufficiently large so that $s > 20|V|$ and $s \geq 20\log(1/\delta)$ where $|V|$ denotes the circuit size of $V$. Pick $r_1, \ldots r_s$ uniformly at random, and share the random bits with the prover. Define a new circuit $W$ by

$$W(x, y_1, \ldots, y_s) := \operatorname*{majority}_{i\in[s]} V(x, y_i, r_i).$$

   In what follows, we call $\bar{y} := (y_1, \ldots, y_s)$ a certificate for $W$.

2. (Verifier) Let $K := m^2(1/\delta)^2 \log(m/\delta)$. Run $C$ independently $K$ times and obtain queries $(q_{k1}, \ldots, q_{km})$ for each $k$th run of $C$ ($k \in [K]$). Send these queries to the prover.

3. (Prover) For each $(k,i) \in [K] \times [m]$, send a certificate $\bar{y}_{ki}$ for $W$; an honest prover sends, if any, some certificate $\bar{y}_{ki}$ such that $W(q_{ki}, \bar{y}_{ki}) = 1$.

4. (Verifier) Let $a^*_{ki} := W(q_{ki}, \bar{y}_{ki}) \in \{0,1\}$ for each $(k,i) \in [K] \times [m]$. Verify that

$$\sum_{\substack{1\leq k\leq K\\ 1\leq i\leq m}} a^*_{ki} \;\geq\; mp^*K \;-\; m\left((\epsilon_1+\epsilon_2)K + \sqrt{K\log(m/\delta)}\right), \tag{5.2}$$

   and if not, output $\perp$ and halt. Otherwise, pick $k \sim [K]$ uniformly at random and output the $k$th run of the reduction of $C$ assuming that the answers from the oracle are $(a_{k1}, \ldots, a_{km})$.

---

**Theorem 5.8** (Correctness of the Generalized Feigenbaum-Fortnow Protocol)**.** *Suppose that the protocol above is given inputs satisfying the promise listed above. Then, the protocol satisfies the completeness and soundness for error* $\epsilon := \epsilon_0 + 2m\epsilon_1 + 3m\epsilon_2 + 3\delta$*, described below:*

- *(Completeness) There exists a prover such that the verifier outputs a with probability at least* $1 - \epsilon$*.*
- *(Soundness) For any prover, the verifier outputs a or* $\perp$ *with probability at least* $1 - \epsilon$*.*

We prove this theorem by a sequence of claims below. The following claim follows from a standard fact about amplification of the success probability of a randomized machine.

[4] As a circuit $V$ outputs "undefined" if an input $(q,y,r)$ is too long, the sets $V_{\text{YES}}, V_{\text{NO}}$ of strings are finite.

**Claim 5.9** (Amplification and Adleman's trick)**.** *With probability at least* $1-\delta$ *over the choice of* $r_1, \ldots, r_s$*, the following holds. For any query* $q \in V_{\text{YES}} \cup V_{\text{NO}}$*,*

- *if* $q \in V_{\text{YES}}$ *then* $W(q, \bar{y}) = 1$ *for some certificate* $\bar{y} := (y_1, \ldots, y_s)$*, and*
- *if* $q \in V_{\text{NO}}$ *then* $W(q, \bar{y}) = 0$ *for any certificate* $\bar{y} := (y_1, \ldots, y_s)$*.*

*Proof.* First, note that $|V_{\text{YES}} \cup V_{\text{NO}}|$ is at most $2^{s/20}$. Indeed, the circuit size of $V$ is less than $s/20$, and hence for any input $q$ of length $\geq s/20$, $V$ is not defined; thus $q \notin V_{\text{YES}} \cup V_{\text{NO}}$. That is, the length of every query in $V_{\text{YES}} \cup V_{\text{NO}}$ is less than $s/20$.

Fix any $q$ such that $q \in V_{\text{YES}}$ or $q \in V_{\text{NO}}$; in the former case, let $a_q := 1$ and $a_q := 0$ otherwise. Our claim is that $\max_{\bar{y}} W(q, \bar{y}) = a_q$. For each $i \in [s]$, let $X_i \in \{0,1\}$ be the random variable (over the random choice of $r_1, \ldots, r_s$) such that $X_i := 1$ iff $V(q, y_i, r_i) = 1$ for some $y_i$; in other words, $X_i := \max_{y_i} V(q, y_i, r_i) \in \{0,1\}$. Observe that

$$\max_{\bar{y}} W(q, \bar{y}) = \max_{\bar{y}} \operatorname*{majority}_{i \in [s]} V(q, y_i, r_i) = \operatorname*{majority}_{i \in [s]} \max_{y_i} V(q, y_i, r_i) = \operatorname*{majority}_{i \in [s]} X_i.$$

By the assumption on $V$, we have $|\mathbb{E}[X_i] - a_q| \leq \frac{1}{4}$ for any $i \in [s]$; hence, $\text{majority}_{i \in [s]} X_i \neq a_q$ implies $|\frac{1}{s}\sum_i (X_i - \mathbb{E}[X_i])| \geq \frac{1}{4}$. By Hoeffding's inequality (Lemma 5.4),

$$\Pr[\max_{\bar{y}} W(q, \bar{y}) \neq a_q] \leq \Pr\left[\left|\sum_{i=1}^{s} (X_i - \mathbb{E}[X_i])\right| \geq \frac{s}{4}\right] \leq 2\exp\left(-2s\,(1/4)^2\right) \leq 2^{-s/10}.$$

Now, by the union bound over all $q \in V_{\text{YES}} \cup V_{\text{NO}}$, the probability that there exists some $q \in V_{\text{YES}} \cup V_{\text{NO}}$ such that $\max_{\bar{y}} W(q, \bar{y}) \neq a_q$ is at most $|V_{\text{YES}} \cup V_{\text{NO}}| \cdot 2^{-s/10} \leq 2^{-s/20} \leq \delta$. □

For each $(k, i) \in [K] \times [m]$, define $a_{ki} \in \{0,1\}$ as $a_{ki} := 1$ if and only if $W(q_{ki}, \bar{y}) = 1$ for some certificate $\bar{y}$. The honest prover sends a certificate for $W$ (if any), and thus $a_{ki} = a^*_{ki}$; on the other hand, when communicating with a cheating prover, we have only $a^*_{ki} \leq a_{ki}$. The next claim shows the sum of $(a_{ki})$ concentrates around its mean.

**Claim 5.10** (Concentration)**.** *Under the event of Claim 5.9, with probability at least* $1-\delta$*, the following holds:*

$$\left|\sum_{\substack{1 \leq k \leq K \\ 1 \leq i \leq m}} a_{ki} - mKp^*\right| \leq m\left((\epsilon_1 + \epsilon_2)K + \sqrt{K \log(m/\delta)}\right).$$

*Proof.* Fix any $i \in [m]$. By the assumption on $C$, the queries $q_{1i}, \ldots, q_{Ki}$ are independent and identically distributed according to $\mathcal{Q}$; hence, $a_{1i}, \ldots, a_{Ki} \in \{0,1\}$ are also independent random variables. The expectation $\mathbb{E}[a_{ki}]$ of these random variables is equal to $\Pr_{q \sim \mathcal{Q}}[W(q, \bar{y}) = 1 \text{ for some } \bar{y}]$.

We claim that, for any $(k, i) \in [K] \times [m]$, the expectation $\mathbb{E}[a_{ki}]$ is equal to $p^*$ up to additive error $\epsilon_1 + \epsilon_2$. Under the event of Claim 5.9, we have $q \in V_{\text{YES}} \implies \exists \bar{y}.\ W(q, \bar{y}) = 1 \implies q \notin V_{\text{NO}}$ for any $q \in \{0,1\}^*$; hence,

$$\Pr[q \in V_{\text{YES}}] \leq \Pr[\exists \bar{y}.\ W(q, \bar{y}) = 1] \leq \Pr[q \notin V_{\text{NO}}]. \tag{5.3}$$

Similarly, since $V_{\text{YES}} \subseteq R \subseteq \{0,1\}^* \setminus V_{\text{NO}}$, we have

$$\Pr[q \in V_{\text{YES}}] \leq \Pr[q \in R] \leq \Pr[q \notin V_{\text{NO}}]. \tag{5.4}$$

Combining (5.3) and (5.4), we obtain

$$|\Pr[q \in R] - \mathbb{E}[a_{ki}]| \leq \Pr[q \notin V_{\text{NO}}] - \Pr[q \in V_{\text{YES}}] \leq \epsilon_2.$$

Therefore, $|\mathbb{E}[a_{ki}] - p^*| \leq |\mathbb{E}[a_{ki}] - \Pr[q \in R]| + |\Pr[q \in R] - p^*| \leq \epsilon_2 + \epsilon_1$.

By Hoeffding's inequality (Lemma 5.4), for each $i \in [m]$,

$$\Pr\left[\left|\sum_{k=1}^{K}(a_{ki} - \mathbb{E}[a_{ki}])\right| \geq \sqrt{K\log(m/\delta)}\right] \leq 2\exp(-2K\log(m/\delta)/K) \leq \delta/m.$$

By the union bound over all $i \in [m]$, with probability at least $1-\delta$, we have

$$\begin{aligned}\left|\sum_{i=1}^{m}\sum_{k=1}^{K} a_{ki} - mKp^*\right| &\leq \left|\sum_{i=1}^{m}\sum_{k=1}^{K}(a_{ki} - \mathbb{E}[a_{ki}])\right| + \left|\sum_{i=1}^{m}\sum_{k=1}^{K}(\mathbb{E}[a_{ki}] - p^*)\right| \\ &\leq \sum_{i=1}^{m}\left|\sum_{k=1}^{K}(a_{ki} - \mathbb{E}[a_{ki}])\right| + mK(\epsilon_1+\epsilon_2) \\ &\leq m\sqrt{K\log(m/\delta)} + mK(\epsilon_1+\epsilon_2).\end{aligned}$$

□

Now we are ready to bound the probability of completeness and soundness. Let $E$ denote any event (which is supposed to be the event that completeness or soundness does not hold); using Claim 5.9 and 5.10, we will bound the probability in the following way:

$$\begin{aligned}\Pr[E] \;\leq\; 2\delta + \Pr[\, E \wedge &(\text{the event of Claim 5.9 holds}) \\ &\wedge (\text{the concentration of Claim 5.10 occurs})\,].\end{aligned}$$

That is, assuming that the events of Claim 5.9 and 5.10 happens, we will analyze the probability of completeness and soundness.

**Claim 5.11** (Completeness)**.** *The verifier outputs $a$ with probability at least $1-\epsilon$ when interacting with the honest prover.*

*Proof.* The verifier *does not* output $a$ only if

- the inequality (5.2) does not hold, or
- for a random $k \sim [K]$, the $k$th run of $C$ is not correct.

The honest prover sets $a^*_{ki} := a_{ki}$ for any $(k,i) \in [K] \times [m]$. Thus, under the assumption that the concentration of Claim 5.10, the inequality (5.2) is satisfied; that is, the verifier does not output $\perp$, and hence it remains to bound the probability that, for a random $k \sim [K]$, the $k$th run of $C$ is not correct.

The $k$th run of $C$ is not correct only if the reduction itself makes a mistake, or there exists some $i \in [m]$ such that $a^*_{ki} \neq R(q_{ki})$ (which happens only if $q_{ki} \notin V_{\text{YES}} \cup V_{\text{NO}}$ for the honest prover). The former probability is at most $\epsilon_0$, and the latter is at most $m\epsilon_2$.

Overall, the verifier outputs $a$ with probability at least $1-2\delta-\epsilon_0-m\epsilon_2 \geq 1-\epsilon$.

□

**Claim 5.12** (Soundness)**.** *For any cheating prover, the verifier outputs $a$ or $\perp$ with probability at least $1-\epsilon$.*

*Proof.* The verifier outputs the wrong answer $1-a$ only if for a random $k \sim [K]$, the $k$th run of $C$ is not correct.

Recall that we have $a^*_{ki} \le a_{ki}$ for any $(k,i) \in [K] \times [m]$ no matter how a prover tries to cheat. The main difference between the proof of Claim 5.11 is that, for a random choice $k \sim [K]$ of the verifier, a prover may be cheating so that $a^*_{ki} < a_{ki}$ for some $i \in [m]$; as a consequence, the $k$th run of $C$ is more likely to be wrong. On the other hand, the number of $(k,i) \in [K] \times [m]$ such that $a^*_{ki} < a_{ki}$ is small: Indeed, under the event that the verifier does not output $\perp$, the inequality (5.2) holds, and we also have the concentration of Claim 5.10; hence, we obtain $\sum_{k=1}^{K} \sum_{i=1}^{m} (a_{ki} - a^*_{ki}) \le 2m \left( (\epsilon_1 + \epsilon_2)K + \sqrt{K \log(m/\delta)} \right)$. Thus, the probability that $a^*_{ki} < a_{ki}$ for some $i \in [m]$ over the random choice of $k \sim [K]$ is at most $2m(\epsilon_1+\epsilon_2) + m\sqrt{\log(m/\delta)/K} \le 2m(\epsilon_1+\epsilon_2) + \delta$.

Overall, the probability that the verifier outputs the wrong answer is at most $(2\delta + \epsilon_0 + m\epsilon_2) + (2m(\epsilon_1+\epsilon_2) + \delta) \le \epsilon$. □

*Proof of Theorem 5.8.* Immediate from Claim 5.11 and 5.12. □

**Remark.** The generalized Feigenbaum-Fortnow protocol above also works for any reduction that does not necessarily output a Boolean value (e.g., a reduction solving a search problem), with a suitable modification on the completeness and soundness.

## 5.6 Simulating Long Queries by $\mathsf{AM} \cap \mathsf{coAM}$

Using the generalized Feigenbaum-Fortnow protocol, we prove our main result:

**Theorem 5.13** (Restatement of Theorem 5.1)**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$ be any (not necessarily computable) description interpreter and $\gamma \colon \mathbb{N} \to [0,1)$ be a parameter such that*

- *there exists a constant $\epsilon > 0$ such that $s(\ell) \le (1-\epsilon)\ell$ for all large $\ell \in \mathbb{N}$, and*
- *there exists a constant $c > 0$ such that $\gamma(\ell) \le 1 - \ell^{-c}$ for all large $\ell \in \mathbb{N}$.*

*Then,*

$$\bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPP}^R_{\|} \quad \subseteq \quad \mathsf{NP/poly} \cap \mathsf{coNP/poly} \cap \mathsf{S}_2^{\mathsf{NP}}.$$

*Moreover, if $G$ can be computed in $2^{O(\ell)}$ time, then we also have*

$$\bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPP}^R_{\|} \quad \subseteq \quad \mathsf{AM} \cap \mathsf{coAM}.$$

As shown in Section 5.4, reductions that make only short queries can be simulated in $\mathsf{S}_2^{\mathrm{p}}$. On the other hand, in this section, we show that reductions that make only long queries can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$. The advice in Theorem 5.13 is used in order to give the characteristic function of $\mathrm{Im}(G)$ for all strings of length $O(\log n)$, under which situation the rest of reductions can be simulated in $\mathsf{AM} \cap \mathsf{coAM} \subseteq \mathsf{NP/poly} \cap \mathsf{coNP/poly}$. If a hitting set generator is

computable in exponential time, then the advice can be computed in polynomial time and thus can be removed. Without any advice and without any computational bound on a hitting set generator, the reduction can be simulated in $\mathsf{S}_2^{\mathsf{NP}}$, which is a complexity class that can simulate $\mathsf{AM} \cap \mathsf{coAM}$ and $\mathsf{S}_2^{\mathrm{P}}$.

Therefore, the main ingredient of the main theorem is to simulate long queries in $\mathsf{AM} \cap \mathsf{coAM}$:

**Theorem 5.14** ($\mathsf{AM}$ Simulation of Long Queries)**.** *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$ be any description interpreter. Let $t, \theta, \alpha_0 \colon \mathbb{N} \to \mathbb{N}$ be efficiently computable functions. Suppose that there exists a randomized $t(n)$-time nonadaptive black-box reduction $M$ from a language $L$ to any $\gamma$-avoiding oracle for $G$ such that the length of any query that $M$ makes on input length $n$ is at least $\theta(n)$. Suppose also that, for all large $n \in \mathbb{N}$,*

- $\alpha_0(n) \le \frac{1}{16\mathrm{e}^3 t(n)^2} 2^{\theta(n) - s(\theta(n))}$, *and*
- $\alpha_0(n) \cdot \big(1 - 2^{s(\ell)-\ell} - \gamma(\ell)\big) \ge 1$ *for all $\ell \in \mathbb{N}$ such that $\theta(n) \le \ell \le t(n)$.*

*Then, there exists an $\mathsf{AM} \cap \mathsf{coAM}$ protocol running in $t(n)^{O(1)}$ time that decides $L$.*

We first observe that this is sufficient to prove Theorem 5.13.

*Proof of Theorem 5.13 from Theorem 5.14.* Take any language $L \in \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPP}_{\|}^R$. By Proposition 5.6, we have a randomized $t(n)$-time nonadaptive black-box reduction $M$ from $L$ to any $\gamma$-avoiding oracle for $G$, where $t(n) = n^{O(1)}$. By the assumption on the seed length $s$, we have $\epsilon\ell \le \ell - s(\ell)$ for all large $\ell \in \mathbb{N}$. For any $\ell \in \mathbb{N}$ between $\theta(n)$ and $t(n)$, we have $1 - 2^{s(\ell)-\ell} - \gamma(\ell) \ge \ell^{-c} - 2^{-\epsilon\ell} \gg \ell^{-c}/2$; hence, by defining $\alpha_0(n) := 2t(n)^c$, we obtain $\alpha_0(n) \ge \big(1 - 2^{s(\ell)-\ell} - \gamma(\ell)\big)^{-1}$ for all $\ell$ between $\theta(n) \le \ell \le t(n)$. On the other hand, $2^{\theta(n) - s(\theta(n))}/t(n)^2 \ge 2^{\epsilon\theta(n)}/t(n)^2$; thus, for $\theta(n) := ((c+2+1)\log t(n))/\epsilon$, we obtain $\alpha_0(n) \ll 2^{\theta(n)-s(\theta(n))}/16\mathrm{e}^3 t(n)^2$ for all large $n$.

The assumptions about parameters of Theorem 5.14 are thus satisfied for $\theta(n) = O(\log t(n))$. In particular, we can encode the characteristic function of the set $\cup_{\ell \le \theta(n)} \mathrm{Im}(G_\ell)$ as an advice string of length $t(n)^{O(1)}$. Given such an advice, we can modify the reduction $M$ so that $M$ does not make any query $q$ of length at most $\theta(n)$: Indeed, if the original reduction makes a query $q$ of length $\le \theta(n)$, then we modify the reduction so that $q$ is answered according to whether $q \in \{0,1\}^* \setminus \mathrm{Im}(G_{|q|})$, which can be decided by using the advice. After this modification, by using Theorem 5.14, $M$ can be simulated in $\mathsf{AM} \cap \mathsf{coAM}$. We thus obtain $L \in \mathsf{AM/poly} \cap \mathsf{coAM/poly} = \mathsf{NP/poly} \cap \mathsf{coNP/poly}$.

Moreover, if $G$ is computable in $2^{O(\ell)}$, then the advice can be computed in polynomial time: Indeed, by an exhaustive search, one can compute $\mathrm{Im}(G_\ell)$ in time $2^{O(s(\ell))} \cdot 2^{O(\ell)} = 2^{O(\ell)} \le t(n)^{O(1)}$ for all $\ell \le \theta(n)$.

Finally, we sketch an $\mathsf{S}_2^{\mathsf{NP}}$ algorithm for deciding $L$ when $G$ is not necessarily computable and no advice is given: As in Theorem 5.7, each competitive prover sends a verifier all the strings in $\mathrm{Im}(G)$ of length at most $\theta(n)$. Let $I_0, I_1 \subseteq \{0,1\}^*$ be the claimed image of $G$. Then we modify the reduction $M$ so that any short query is answered according to $\{0,1\}^* \setminus I_0 \setminus I_1$. Now by applying Theorem 5.14, we obtain an $\mathsf{AM}$ algorithm deciding $L$. In particular, there exists an $\mathsf{NP}$ machine $V$ such that $\Pr_r[V(x,r) = L(x)] \ge \frac{2}{3}$ for every input $x$. This randomized computation can be derandomized as in Theorem 5.7, by requesting

each competitive prover to send a sequence of coin flips $r_1, \cdots, r_s$. Thus we obtain $L \in \mathsf{S}_2^{\mathsf{NP}}$. □

In the rest of this section, we show how to simulate long queries by a constant-round interactive proof system (i.e., a proof of Theorem 5.14). For simplicity, we focus on the case when $t(n) = n^{O(1)}$. Let $M$ be a randomized $t(n)$-time nonadaptive black-box reduction to any $\gamma$-avoiding oracle for $G$. Let $x \in \{0,1\}^*$ be an input of length $n$.

We first modify the reduction so that we can assume useful properties. By the modifications explained in Section 5.3, we may assume that the number of queries that $M$ makes is exactly $m(n)$ $(\leq t(n))$ on inputs of length $n$. We may also assume that each query of $M$ is identically distributed; Let $\mathcal{Q}_x$ be the query distribution of $M$ on input $x$.

As explained in the introduction, one of the keys of our proof is that we can replace a $\gamma$-avoiding oracle for $G$ by an oracle defined based only on the query distribution $\mathcal{Q}_x$. Here we introduce such oracles and justify the replacement. For any $\alpha > 0$, define $L_\alpha, H_\alpha$ and $R_\alpha$ by

$$\begin{aligned} L_\alpha &:= \{\, q \in \{0,1\}^* \mid q \text{ is } \alpha\text{-light with respect to } \mathcal{Q}_x \,\}, \\ H_\alpha &:= \{0,1\}^* \setminus L_\alpha = \{\, q \in \{0,1\}^* \mid q \text{ is } \alpha\text{-heavy with respect to } \mathcal{Q}_x \,\}, \text{ and} \\ R_\alpha &:= L_\alpha \setminus \mathrm{Im}(G). \end{aligned}$$

For large enough $\alpha > 0$, we can easily show that $R_\alpha$ $\gamma$-avoids $G$.

**Claim 5.15.** *For any $\gamma\colon \mathbb{N} \to [0,1)$ and $\alpha > 0$ and for any length $\ell \in \mathbb{N}$, if*

$$\gamma(\ell) + 1/\alpha \leq 1 - 2^{s(\ell)-\ell},$$

*then $R_\alpha$ is a $\gamma$-avoiding set at length $\ell$ for $G$.*

*Proof.* Since $R_\alpha \subseteq \{0,1\}^* \setminus \mathrm{Im}(G)$, it suffices to show that $\Pr_{w\sim\{0,1\}^\ell}[w \notin R_\alpha] \leq 1-\gamma(\ell)$. Note that $w \notin R_\alpha$ if either $w \in \mathrm{Im}(G_\ell)$ or $w$ is $\alpha$-heavy. The probability of the former case is at most $2^{-\ell} \cdot |\mathrm{Im}(G_\ell)| \leq 2^{s(\ell)-\ell}$. Similarly, the probability of the latter case is bounded above by $2^{-\ell} \cdot |H_\alpha^{=\ell}|$, where $H_\alpha^{=\ell} = \{\, q \in \{0,1\}^\ell \mid q \text{ is } \alpha\text{-heavy} \,\}$. On the other hand, we have

$$|H_\alpha^{=\ell}| \cdot \alpha 2^{-\ell} \;\leq\; \sum_{q \in H_\alpha^{=\ell}} \Pr_{w\sim\mathcal{Q}_x}[\, w = q \,] \;=\; \Pr_{w\sim\mathcal{Q}_x}[\, w \in H_\alpha^{=\ell} \,] \;\leq\; 1.$$

Hence, $|H_\alpha^{=\ell}| \leq 2^\ell/\alpha$. Thus,

$$\Pr_{w\sim\{0,1\}^\ell}[w \notin R_\alpha] \;\leq\; 2^{s(\ell)-\ell} + 1/\alpha \;\leq\; 1 - \gamma(\ell),$$

proving that $\Pr_{w\sim\{0,1\}^\ell}[w \in R_\alpha] \geq \gamma(\ell)$. □

Since the reduction $M$ does not make any query $q$ such that $|q| \notin [\theta(n), t(n)]$, Claim 5.15 guarantees that the reduction $M$ works by using $R_\alpha$ on inputs $x$ of length $n$ if $\gamma(\ell) + 1/\alpha \leq 1 - 2^{s(\ell)-\ell}$ for all $\ell \in \mathbb{N}$ such that $\theta(n) \leq \ell \leq t(n)$. As this condition is satisfied by our assumptions of Theorem 5.14 for any $\alpha \geq \alpha_0(n)$ and any input $x$ of length $n$, we have

$$\Pr_M\left[M^{R_\alpha}(x) = L(x)\right] \;\geq\; \frac{15}{16}. \tag{5.5}$$

On the other hand, we can show below that $M(x)$ cannot distinguish $R_\alpha$ and $L_\alpha$ when $\alpha$ is small enough.

**Claim 5.16.** *For any $\alpha > 0$ and input $x \in \{0,1\}^*$ of length $n$, and for $\epsilon := \alpha 2^{s(\theta(n))-\theta(n)} \cdot m(n)t(n)$,*

$$\Pr_M \left[ M^{L_\alpha}(x) \neq M^{R_\alpha}(x) \right] \ \leq \ \epsilon. \tag{5.6}$$

*Proof.* Recall that $R_\alpha = L_\alpha \setminus \mathrm{Im}(G)$. Thus, $M(x)$ may find the difference between $L_\alpha$ and $R_\alpha$ only if it makes a query in $L_\alpha \cap \mathrm{Im}(G)$ in one of its $m(n)$ nonadaptive queries. This probability is at most $m(n) \cdot \Pr_{w \sim \mathcal{Q}_x}[w \in L_\alpha \cap \mathrm{Im}(G)]$ by a union bound.

Here we have

$$\begin{aligned} \Pr_{w \sim \mathcal{Q}_x} [\, w \in L_\alpha \cap \mathrm{Im}(G) \,] \ &= \ \sum_{q \in L_\alpha \cap \mathrm{Im}(G)} \Pr_{w \sim \mathcal{Q}_x} [\, q = w \,] \\ &\leq \ \sum_{q \in \mathsf{supp}(\mathcal{Q}_x) \cap \mathrm{Im}(G)} \alpha \cdot 2^{-|q|}, \end{aligned}$$

where $\mathsf{supp}(\mathcal{Q}_x)$ is the set of all possible queries asked by $M(x)$. By our assumption on $M$, we have $\theta(n) \leq |q| \leq t(n)$ for any $q \in \mathsf{supp}(\mathcal{Q}_x)$. Then it follows

$$\sum_{q \in \mathsf{supp}(\mathcal{Q}_x) \cap \mathrm{Im}(G)} \alpha \cdot 2^{-|q|} \ \leq \ \sum_{\theta(n) \leq \ell \leq t(n)} \alpha \cdot 2^{-\ell} \cdot |\mathrm{Im}(G_\ell)| \ \leq \ \sum_{\theta(n) \leq \ell \leq t(n)} \alpha \cdot 2^{s(\ell)-\ell}$$

because $|\mathrm{Im}(G_\ell)| \leq 2^{s(\ell)}$.

Since we assumed that $\ell - s(\ell)$ is nondecreasing for $\ell \in \mathbb{N}$, we have

$$\sum_{\theta(n) \leq \ell \leq t(n)} \alpha 2^{s(\ell)-\ell} \ \leq \ t(n) \cdot \alpha 2^{s(\theta(n))-\theta(n)} \ = \ \epsilon / m(n).$$

This bound is sufficient to get the desired error bound. □

By Claim 5.16 and our assumptions on $\alpha_0(n)$ of Theorem 5.14, for any $\alpha \leq \mathrm{e}^3 \alpha_0(n)$, we have

$$\Pr_M \left[ M^{L_\alpha}(x) \neq M^{R_\alpha}(x) \right] \leq \frac{1}{16}. \tag{5.7}$$

From the inequalities (5.5) and (5.7), we immediately obtain the following:

**Corollary 5.17.** *For any input $x \in \{0,1\}^*$ of length $n$ and any $\alpha \in [\alpha_0(n), \mathrm{e}^3 \alpha_0(n)]$,*

$$\Pr_M \left[ M^{L_\alpha}(x) = L(x) \right] \ \geq \ \frac{7}{8}.$$

In light of this, our task is now to simulate $M^{L_\alpha}(x)$ for some $\alpha \in [\alpha_0(n), \mathrm{e}^3 \alpha_0(n)]$ (in fact, we will choose the threshold $\alpha$ randomly, as explained later). To this end, we combine the generalized Feigenbaum-Fortnow protocol, the lower bound protocol of Goldwasser and Sipser [GS86], and the heavy-sample protocol of Bogdanov and Trevisan [BT06b]. Here we review the last two protocols. Since these protocols are explained carefully and in detail in the paper [BT06b], we simply review their specifications and use them as a black-box tool.

**Lower Bound Protocol.** Recall that $q \notin L_\alpha$ if and only if $q$ is $\alpha$-heavy. The lower bound protocol of Goldwasser and Sipser [GS86] can be used to give an $\mathsf{AM}$-type witness to any $\alpha$-heavy instance. It is an $\mathsf{AM}$ protocol for showing that a given set of strings has more than $s$ elements for a given threshold $s$. The specification of the lower bound protocol is as follows.

**Inputs.** A set of strings is given as a circuit $C$ on $\{0,1\}^m$, which specifies the set as $C^{-1}(1) := \{\, r \in \{0,1\}^m \mid C(r) = 1 \,\}$. A threshold $s \in \mathbb{N}$ such that $0 \leq s \leq 2^m$. Parameters $\delta, \epsilon \in [0,1]$ represented in unary.

**Promise.**

- Yes instances: $|C^{-1}(1)| \geq s$.
- No instances: $|C^{-1}(1)| \leq (1-\epsilon)s$.

**Sketch of the Protocol.**

1. (Verifier) Send a random hash function $h : \{0,1\}^m \to \{0,1\}^{m'}$ for some appropriate parameter $m'$.
2. (Prover) Send a string $y \in \{0,1\}^m$ claiming that $y \in C^{-1}(1)$ and $h(y) = 0^{m'}$ hold. (Such an $y$ is called *an* $\mathsf{AM}$*-type witness.*)
3. (Verifier) Check the correctness of the prover's claim on the witness $y$.

---

**Theorem 5.18** (Correctness of the lower bound protocol [GS86]; see [BT06b] for a proof)**.** *The lower bound protocol stated above satisfies the following:*

- *(Completeness) Given an yes instance, there exists a prover that makes the verifier accept with probability at least* $1 - \delta$.
- *(Soundness) Given a no instance, for any prover, the verifier accepts with probability at most* $\delta$.

By using the protocol above, it is easy to construct an $\mathsf{AM}$ verifier $V$ that checks whether a given query $q$ is $\alpha$-heavy.

**Claim 5.19.** *For any parameter* $\epsilon(n) \geq 1/\mathsf{poly}(n)$*, there exists an* $\mathsf{AM}$ *verifier* $V$ *such that, for any input* $x \in \{0,1\}^*$ *of length* $n$ *and any query* $q \in \{0,1\}^*$*,*

1. *if* $q$ *is* $\alpha$*-heavy with respect to* $\mathcal{Q}_x$*, then* $\Pr_h[V(x,q,h,y) = 1 \text{ for some } y] \geq \frac{3}{4}$*, and*
2. *if* $q$ *is* $(1-\epsilon(n))\alpha$*-light with respect to* $\mathcal{Q}_x$*, then* $\Pr_h[V(x,q,h,y) = 1 \text{ for some } y] \leq \frac{1}{4}$.

*Proof.* Let $Q_x$ be the circuit that samples the query distribution $\mathcal{Q}_x$; that is, on input $r \in \{0,1\}^m$, the circuit $Q_x(r)$ outputs $q$ so that $\Pr_{r \sim \{0,1\}^m}[Q_x(r) = q] = \Pr_{w \sim \mathcal{Q}_x}[w = q]$ for any $q \in \{0,1\}^*$. Given a string $q \in \{0,1\}^*$ as input, construct a circuit $C_q$ such that $C_q(r) := 1$ iff $Q_x(r) = q$, on input $r \in \{0,1\}^m$. Now use the lower bound protocol for the circuit $C_q$, the threshold $s := \alpha 2^{-|q|} 2^m$, and parameters $\delta := \frac{1}{4}$ and $\epsilon := \epsilon(n)$. The lower bound protocol gives an $\mathsf{AM}$ certificate for the yes instances such that $|C_q^{-1}(1)| \geq s$, which is equivalent to saying that $\Pr_r[Q_x(r) = q] \geq \alpha 2^{-|q|}$, that is, $q$ is $\alpha$-heavy. On the other hand, if $q$ is $(1-\epsilon)\alpha$-light, then we have $|C_q^{-1}(1)| < (1-\epsilon)s$; thus with high probability there is no $\mathsf{AM}$-type witness by the correctness of the lower bound protocol (Theorem 5.18). □

Note that there is a gap between yes instances and no instances; that is, if the probability that $q$ is sampled from $\mathcal{Q}_x$ is between $\alpha$ and $(1-\epsilon)\alpha$, then the behavior of the lower bound protocol is undefined. To circumvent this, we pick the threshold $\alpha$ randomly in the same way with Bogdanov and Trevisan

(cf. [BT06b, Claim 3.2]): Consider the following set $\mathcal{A}_{\alpha_0,\epsilon}$ of thresholds defined by parameters $\alpha_0, \epsilon > 0$, and choose the threshold $\alpha$ uniformly at random from $\mathcal{A}_{\alpha_0,\epsilon}$.

$$\mathcal{A}_{\alpha_0,\epsilon} := \{\, \alpha_0(1+3\epsilon)^i \mid 0 \le i \le 1/\epsilon \,\}.$$

Observe that $\mathcal{A}_{\alpha_0,\epsilon} \subseteq [\alpha_0, \mathrm{e}^3\alpha_0]$. Moreover, the following holds.

**Lemma 5.20.** *For every $\alpha_0 > 0$ and $0 < \epsilon < \frac{1}{3}$ and any constant $c > 0$, and for any distribution $\mathcal{Q}$, with probability at least $1 - 1/c$ over the choice of $\alpha \sim \mathcal{A}_{\alpha_0,\epsilon}$,*

$$\Pr_{q\sim\mathcal{Q}}\left[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\right] \le c\epsilon. \tag{5.8}$$

(Recall that $\mathcal{Q}(q) := \Pr_{w\sim\mathcal{Q}}[w = q]$.)

*Proof.* For any $\epsilon \in \left(0, \frac{1}{3}\right)$ and $q \in \{0,1\}^*$, the intervals $((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})$ are pairwise disjoint for all $\alpha \in \mathcal{A}_{\alpha_0,\epsilon}$; hence for any real $p \in \mathbb{R}$, the probability that $p \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})$ is at most $1/|\mathcal{A}_{\alpha_0,\epsilon}| \le \epsilon$ over the choice of $\alpha \sim \mathcal{A}_{\alpha_0,\epsilon}$. In particular, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\alpha\sim\mathcal{A}_{\alpha_0,\epsilon}}\left[\Pr_{q\sim\mathcal{Q}}\left[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\right]\right] \\
&= \mathop{\mathbb{E}}_{q\sim\mathcal{Q}}\left[\Pr_{\alpha\sim\mathcal{A}_{\alpha_0,\epsilon}}\left[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\right]\right] \;\le\; \epsilon.
\end{aligned}$$

Therefore, by Markov's inequality, the probability that $\Pr_{q\sim\mathcal{Q}}\left[\mathcal{Q}(q) \in ((1\pm\epsilon)\alpha 2^{-|q|})\right] \ge c\epsilon$ is at most $\epsilon/(c\epsilon) = 1/c$. □

In our simulation protocol for $M$, we start with picking $\alpha \sim \mathcal{A}_{\alpha_0,\epsilon}$ randomly. By Lemma 5.20, except for probability $1/O(1)$, the heaviness of almost all queries $q$ sampled from $\mathcal{Q}_x$ is not close to the threshold $\alpha$. As a consequence, the distributional problem $(L_\alpha, \mathcal{Q}_x)$ is solvable by coAM on average; indeed, with probability at least $1-O(\epsilon)$ over the choice of $q \sim \mathcal{Q}_x$, the lower bound protocol of Claim 5.19 solves $L_\alpha$.

**Heavy-sample Protocol.** Next we review the heavy-sample protocol of Bogdanov and Trevisan [BT06b], which is an AM protocol for estimating $\Pr_{q\sim\mathcal{Q}_x}[q \text{ is } \alpha\text{-heavy}]$.

---

**Inputs.** A circuit $Q$ which samples a string according to a distribution $\mathcal{Q}$ on $\{0,1\}^*$. A probability $p \in [0,1]$ represented in binary. Parameters $c > 0$ and $0 < \epsilon < \frac{1}{3}$ represented in unary. A threshold $\alpha > 0$ represented in binary.

**Promise.**

- Yes instances: $\Pr_{q\sim\mathcal{Q}}[\mathcal{Q}(q) \ge \alpha 2^{-|q|}] = p$.
- No instances: $\left|\Pr_{q\sim\mathcal{Q}}[\mathcal{Q}(q) \ge \alpha 2^{-|q|}] - p\right| > 16c\epsilon$.
- We assume the condition (5.8). That is,

$$\Pr_{q\sim\mathcal{Q}}\left[\mathcal{Q}(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\right] \le c\epsilon.$$

**Sketch of the Protocol.**

1. (Verifier) Generate random queries $q_1, \ldots, q_k$ from the distribution $\mathcal{Q}$ for some sufficiently large $k$, and send these queries to the prover.

2. (Prover) For each query $q_i$, tell the verifier whether $q_i$ is $\alpha$-heavy.

3. (Verifier and Prover) To check the prover's claim, run the lower bound protocol of Goldwasser and Sipser [GS86] and the upper bound protocol of Fortnow [For89] in parallel.

---

**Theorem 5.21** (Correctness of the heavy-sample protocol [BT06b])**.** *The heavy-sample protocol specified above satisfies following:*

- *(Completeness) Given any yes instance, there exists a prover that makes the verifier accept with probability at least* $1 - O(\epsilon)$*.*
- *(Soundness) Given any no instance, for any prover, the verifier accepts with probability at most* $O(\epsilon)$*.*

**Protocol for Simulating** $M^{L_\alpha}(x)$**.** Using the protocols reviewed above, we can now simulate the reduction $M$ to an oracle $L_\alpha$ on input $x \in \{0,1\}^*$. Below we explain how to choose the inputs $(C, V, \delta := \frac{1}{100}, p^*)$ for the generalized Feigenbaum-Fortnow protocol.

The generalized Feigenbaum-Fortnow protocol requires an $\mathsf{AM}$ protocol for solving an oracle on average (instead of $\mathsf{coAM}$). By negating answers from the oracle, we can define a new machine $\overline{M}^X$ which simulates the computation of $M^{\{0,1\}^* \setminus X}$ for any given oracle $X$. We thus use the generalized Feigenbaum-Fortnow protocol for simulating $\overline{M}^{H_\alpha}(x)$ with oracle $H_\alpha := \{0,1\}^* \setminus L_\alpha$, which is the set of $\alpha$-heavy queries with respect to $\mathcal{Q}_x$; more specifically, let $C$ be the circuit that simulates the reduction $\overline{M}$ on input $x$ (where the input $x$ is hardwired into the circuit) and we give the circuit $C$ to the protocol as input.

To solve $H_\alpha$ on average by an $\mathsf{AM}$ protocol, we use the lower bound protocol. That is, we build a circuit $V_x$ that simulates the $\mathsf{AM}$ verifier stated in Claim 5.19 on input $x$ and on all the queries $q \in \{0,1\}^*$ that $M(x)$ can make. Then we give the circuit $V_x$ to the protocol as input.

We also need to give as advice a probability $p^*$ that approximates $\Pr_{q \sim \mathcal{Q}_x}[q \in H_\alpha]$, which can be estimated by using the heavy-sample protocol. We require the prover to send $p^*$ first, and then we verify the prover's claim by running the heavy-sample protocol; if the test passes, then we give $p^*$ to the generalized Feigenbaum-Fortnow protocol as input.

Summarizing the discussion above, our whole simulation algorithm is given below.

---

**Inputs.** A string $x \in \{0,1\}^*$ of length $n$.

**Promise.** Let $\alpha_0 := \alpha_0(n)$. Then, for any $\alpha \in [\alpha_0, \mathrm{e}^3\alpha_0]$, we assume that

$$\Pr_M[M^{L_\alpha}(x) = L(x)] \geq \frac{7}{8},$$

(which is guaranteed by Corollary 5.17).

**Protocol.**

1. (Preprocess) Set an error parameter $\epsilon := 1/c_0 m(n)$ for a sufficiently large constant $c_0$ (represented in unary).

2. (Verifier) Pick a threshold $\alpha \sim \mathcal{A}_{\alpha_0,\epsilon} \subseteq [\alpha_0, \mathrm{e}^3\alpha_0]$ randomly. Send $\alpha$ to the prover.

3. (Prover) Send $p^* \in [0,1]$ to the verifier. An honest prover sends $p^* := \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]$.

4. (Verifier and Prover) Run the heavy-sample protocol in order to verify that $p^* \approx \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]$ for the distribution $\mathcal{Q}_x$ and error parameter $\epsilon$ and parameter $c := 100$; if the test does not pass, output $\perp$ and halt.

5. (Verifier and Prover) Build a circuit $C$ simulating $\overline{M}$ on the hardwired input $x$, and a circuit $V_x$ simulates the $\mathsf{AM}$ verifier for $\alpha$-heaviness. Run the generalized Feigenbaum-Fortnow protocol on input $(C, V_x, \delta := \frac{1}{100}, p^*)$, and output the result of the protocol.

---

Now we argue that our simulation protocol is correct:

**Claim 5.22.** *The simulation protocol stated above satisfies the following: for any $x \in \{0,1\}^*$,*

- *(Completeness) there exists a prover such that the verifier outputs $L(x)$ with probability at least $3/4$, and*
- *(Soundness) for any prover, the verifier outputs $L(x)$ or $\perp$ with probability at least $3/4$.*

*Proof.* Fix any input $x \in \{0,1\}^*$. By Lemma 5.20, with probability at least $1 - \frac{1}{100}$ over the choice of $\alpha \sim \mathcal{A}_{\alpha_0,\epsilon}$, the condition (5.8) holds for $c := 100$ and $\mathcal{Q} := \mathcal{Q}_x$; that is,

$$\Pr_{q\sim\mathcal{Q}_x}\left[\mathcal{Q}_x(q) \in ((1-\epsilon)\alpha 2^{-|q|}, (1+\epsilon)\alpha 2^{-|q|})\right] \leq 100\epsilon.$$

In what follows, we assume this event happens and analyze the probability of the completeness and soundness.

Suppose that a prover sends $p^*$. If the prover is honest then we have $p^* = \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]$. Hence the completeness of the heavy-sample protocol implies that, with probability at least $1 - O(\epsilon) \gg 1 - \frac{1}{100}$, the protocol accepts. On the other hand, if a cheating prover sends $p^*$ such that $|p^* - \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]| > 16c\epsilon$, then with probability at least 0.99 the cheat can be caught by the soundness of the heavy-sample protocol.

Thus, at the point that the generalized Feigenbaum-Fortnow protocol starts, for any prover, with probability at least 0.98, we have $|p^* - \Pr_{q\sim\mathcal{Q}_x}[q \in H_\alpha]| \leq 16c\epsilon$ and moreover the condition (5.8) holds. Under this event, the promise of the generalized Feigenbaum-Fortnow protocol is satisfied:

- Define $a := L(x)$, $\epsilon_0 := \frac{1}{8}$, and $R := H_\alpha$. Then we have $\Pr_C[C^R = a] \geq 1 - \epsilon_0$ by the promise of our simulation protocol (Corollary 5.17).
- The advice $p^*$ satisfies $|p^* - \Pr_{q\sim\mathcal{Q}_x}[q \in R]| \leq \epsilon_1$ for $\epsilon_1 := 16c\epsilon$.
- By Claim 5.19 and (5.8), we have $\Pr_{q\sim\mathcal{Q}_x}[q \notin V_{\mathrm{YES}} \cup V_{\mathrm{NO}}] \leq \Pr_{q\sim\mathcal{Q}_x}[\mathcal{Q}_x(q) \notin (1\pm\epsilon)\alpha 2^{-|q|}] \leq \epsilon_2$ for $\epsilon_2 := 100\epsilon$.

Therefore, from the correctness of the generalized Feigenbaum-Fortnow protocol (Theorem 5.8), our simulation protocol satisfies the completeness and soundness with probability at least $1 - 0.02 - (\frac{1}{8} + 2m(n)\epsilon_1 + 3m(n)\epsilon_2 + 3\delta) \geq \frac{3}{4}$, where the last inequality holds for some large constant $c_0$. □

This completes the proof of Theorem 5.14.

# Chapter 6

# Reductions to the Set of Kolmogorov-Random Strings

Historically, *computability theory* precedes complexity theory, the latter of which is a quantitative version of the former. The fundamental theorem of Turing [Tur36] shows that the halting problem $\mathsf{HALT}$ cannot be solved by a Turing machine, whereas its analogue in complexity theory, i.e., the $\mathsf{P}$ versus $\mathsf{NP}$ question, is wide open. Studying a computability theoretic analogue of MCSP would be the first step towards better understanding of MCSP. We thus investigate what can be reduced to the set of (resource-unbounded) Kolmogorov-random strings, which can be seen as an analogue of MCSP in computability theory. It was conjectured by Allender [All12] and others that the computational power of nonadaptive deterministic polynomial-time reductions to the set of Kolmogorov-random is exactly characterized by $\mathsf{BPP}$, intuitively because nonadaptive deterministic reductions could only make use of Kolmogorov-random strings as a source of pseudorandomness.

In this chapter, we present strong evidence *against* this conjecture by showing that every language in the exponential-time hierarchy is reducible to the set of Kolmogorov-random strings under $\mathsf{PH}$ reductions; in particular, the conjecture is false unless the exponential-time hierarchy collapses to $\mathsf{BPEXP}$. Moreover, our reduction cannot be regarded as a black-box reduction to avoiding hitting set generators (unless the exponential-time hierarchy collapses to the second level), thereby showing that nonadaptive deterministic efficient reductions can exploit the power of Kolmogorov-random strings not just as a distinguisher for hitting set generators.

## 6.1 Background

As shown in Theorem 2.20, there is a clear relationship between Kolmogorov-randomness and pseudorandomness: Any string generated by a computable process has low Kolmogorov complexity, and thus the set of Kolmogorov-random strings serves as a distinguisher for any computable hitting set generator; in particular, Allender, Buhrman, Koucký, van Melkebeek, and Ronneburger [ABK+06b] proved a curious inclusion that $\mathsf{PSPACE} \subseteq \mathsf{P}^{R_{\mathrm{K}_U}}$ for every universal Turing machine $U$. Here, $\mathrm{K}_U(x)$ denotes the Kolmogorov complexity of a string $x$ defined by the universal Turing machine $U$, and $R_{\mathrm{K}_U}$ denotes the set of $(n/2)$-random strings with respect to $\mathrm{K}_U$. Similarly, Allender, Buhrman, Koucký [ABK06a] showed that $\mathsf{NEXP} \subseteq \mathsf{NP}^{R_{\mathrm{K}_U}}$; Buhrman, Fortnow, Koucký, and Loff [BFKL10] showed that $\mathsf{BPP} \subseteq \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$.

Note that since the set of Kolmogorov-random strings is not computable, it

is absurd to hope that efficiently computable complexity classes could be characterized in terms of efficient reductions to $R_{\mathrm{K}_U}$. Nevertheless, it was shown by Cai, Downey, Epstein, Lempp, and Miller [CDE+14] that when the intersection is taken over all prefix universal Turing machines $U$, any language efficiently reducible to $R_{\mathrm{K}_U}$ is decidable, and moreover Allender, Friedman, and Gasarch [AFG13] showed that the upper bound is PSPACE:

**Theorem 6.1** ([BFKL10, AFG13, CDE+14]). $\mathsf{BPP} \subseteq \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{PSPACE}$, *where the intersection is taken over all prefix universal Turing machines.*

For some technical reasons, the results of [AFG13, CDE+14] are known to hold only for *prefix* universal Turing machines. However, a similar upper bound can be obtained for usual universal Turing machines by imposing some constraints on reductions [HK18].

Theorem 6.1 leads us to the following natural question: Is it possible to exactly characterize the computational power of $R_{\mathrm{K}_U}$ under polynomial-time nonadaptive reductions? Intuitively, any polynomial-time nonadaptive reduction cannot make any use of the set of Kolmogorov-random strings of length larger than $O(\log n)$, because the Kolmogorov complexity of any query that the reduction can make on input $0^n$ is at most $O(\log n)$. It was argued in [ABFL14] that, intuitively, short queries to Kolmogorov-random strings could only be used as a source of pseudorandomness. Allender [All12] thus explicitly conjectured that the lower bound of Theorem 6.1 is exactly tight, and since then a fair amount of efforts have been made to verify the conjecture.

**Conjecture 6.2** ([BFKL10, All12, ADF+13, ABFL14, HK18]). $\mathsf{BPP} = \bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$.

Beyond its curiosity, such a characterization might enable us to study the complexity class BPP of languages efficiently solvable by a randomized computation by using the techniques of Kolmogorov complexity. Moreover, Conjecture 6.2 is interesting from the viewpoint of the study of MCSP: It is known that, for any function $f\colon \{0,1\}^n \to \{0,1\}$, the Kolmogorov complexity $\mathrm{K}_U(\mathtt{tt}(f))$ and the size $\mathsf{size}^{\mathsf{HALT}}(f)$ of minimum circuits with oracle access to the halting problem HALT are polynomially related to each other [ABK+06b]; thus $R_{\mathrm{K}_U}$ can be regarded as a computability theoretic analogue of MCSP. In light of this, Conjecture 6.2 implies non-NP-hardness results about $R_{\mathrm{K}_U}$ under nonadaptive polynomial-time reductions (unless $\mathsf{NP} \subseteq \mathsf{BPP}$), and thus studying Conjecture 6.2 would give us some new insights about non-NP-hardness of MCSP, which has been a focus of recent work on MCSP (e.g. [MW17, HW16, HP15, AHK17, AH17]). We refer the reader to the survey of Allender [All12, All17] for more background on Conjecture 6.2.

## 6.2 Evidence against Allender's Conjecture

We disprove Conjecture 6.2 under the assumption that the exponential-time hierarchy does not collapse to BPEXP. Our main new insight is to consider a sparse language in $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}}$, or in other words, an exponential-time analogue of Theorem 6.1. We show that every language in the exponential-time hierarchy EXPH is reducible to $R_{\mathrm{K}_U}$ under PH reductions, and thus:

**Theorem 6.3.** $\mathsf{EXPH} \subseteq \bigcap_U \mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXPSPACE}$, *where the intersection is taken over all prefix universal Turing machines.*

In particular, by a standard padding argument, it implies that $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \neq \mathsf{BPP}$ unless $\mathsf{EXPH} = \mathsf{BPEXP}$. (Indeed, if $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} = \mathsf{BPP}$, then a padding argument implies $\mathsf{EXPH} \subseteq \bigcap_U \mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}} = \mathsf{BPEXP}$.)

We also show that Theorem 6.3 crucially makes use of the fact that the set of Kolmogorov-random strings serves not just as a distinguisher for hitting set generators. Indeed, by using the technique of Chapter 5, the reductions to any dense subset of Kolmogorov-random strings can be simulated by $\mathsf{S}_2^{\mathrm{exp}}$ (and the simulation is tight under some reducibility notion).

**Theorem 6.4** (informal)**.** *Fix any universal Turing machine $U$. Then we have*

$$\bigcap_{R\colon \gamma\text{-avoids } U} \mathsf{EXP}^{R^{\leq \mathsf{poly}}} = \bigcap_{R\colon \gamma\text{-avoids } U} \mathsf{BPEXP}^{R^{\leq \mathsf{poly}}} = \mathsf{S}_2^{\mathrm{exp}}$$

*Here $R^{\leq \mathsf{poly}}$ means that the length of queries is restricted to be at most a polynomial in the input length.*

Theorem 6.4 means that the reduction of Theorem 6.3 cannot be replaced with a reduction to any dense subset of Kolmogorov-random strings unless $\mathsf{EXPH}$ collapses to $\mathsf{S}_2^{\mathrm{exp}}$. In particular, the intuition that nonadaptive polynomial-time reductions could exploit $R_{\mathrm{K}_U}$ only as a source of pseudorandomness is wrong.

### 6.2.1 Proof of Theorem 6.3

Now we present the proof of Theorem 6.3. We note that at the core of Theorem 6.3 is the following reduction from $\mathsf{EXPH}$ to $R_{\mathrm{K}_U}$.

**Theorem 6.5.** $\mathsf{EXPH} \subseteq \mathsf{PH}^{R_{\mathrm{K}_U}}$ *for any universal Turing machine $U$.*

We observe that Theorem 6.5 is enough to prove the main result.

*Proof of Theorem 6.3.* The lower bound follows from Theorem 6.5. For the upper bound, by the results of [CDE+14, AFG13], we have $\bigcap_U \mathsf{P}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{PSPACE}$. By a standard padding argument, we obtain $\bigcap_U \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXPSPACE}$, from which the upper bound follows since $\mathsf{PH}^{R_{\mathrm{K}_U}} \subseteq \mathsf{EXP}_{\|}^{R_{\mathrm{K}_U}}$. □

Now we proceed to a proof of Theorem 6.5. We first observe that, given $\mathsf{HALT}$ as an oracle, one can compute any decidable language with oracle access to $\mathsf{HALT}$ in $\mathsf{P}^{\mathsf{HALT}}$.

**Lemma 6.6.** *Let $k$ be any positive constant. Let $M$ be any oracle machine that, on every input $x$, halts eventually and makes a query of length at most $|x|^k$. Then the language decided by $M$ is in $\mathsf{P}^{\mathsf{HALT}}$.*

*Proof Sketch.* The proof is essentially the same with [ABK+06b, Theorem 27], and thus we just sketch a proof. The idea is to decide $M$ in the following two steps: First, by using a binary search and the oracle access to $\mathsf{HALT}$, one can decide the number of all the strings in $\mathsf{HALT}$ of length at most $n^k$ in polynomial time. Then, given the number of strings in $\mathsf{HALT}$ of length at most $n^k$ as advice, the computation of $M^{\mathsf{HALT}}$ becomes now computable, and hence it reduces to $\mathsf{HALT}$. □

We also recall that the halting problem is reducible to the set of random strings under $\mathsf{P}/\mathsf{poly}$ reductions, which was established by exploiting the fact that the set of random strings can be distinguisher for any computable pseudorandom generator.

**Theorem 6.7** ([ABK+06b]). $\mathsf{HALT} \in \mathsf{P}^{R_{\mathrm{K}_U}}/\mathsf{poly}$ *for any universal Turing machine* $U$.

While the ingredients above are enough to obtain $\mathsf{EXP}$ reductions, in order to obtain $\mathsf{PH}$ reductions, we make use of the efficient proof system of $\mathsf{PH}$ given by Kiwi, Lund, Spielman, Russell, and Sundaram [KLS+00]. For simplicity, we state their results in the case of the number of alternation is 2, but their results hold for every constant number of alternation. We also state their results in terms of $\Sigma_2^{\mathsf{EXP}}$ instead of $\Sigma_2^{\mathsf{P}}$.

**Theorem 6.8** (Kiwi, Lund, Spielman, Russell, and Sundaram [KLS+00]). *For every language* $L$ *in* $\Sigma_2^{\mathsf{EXP}}$*, there exists a randomized polynomial-time verifier such that,*

1. *for every input* $x \in L$*, there exists an oracle* $A$ *such that for any oracle* $B$*,* $V^{A,B}(x)$ *accepts with probability 1, and*
2. *for every input* $x \notin L$*, for all oracles* $A$*, there exists an oracle* $B$*,* $V^{A,B}(x)$ *accepts with probability at most* $\frac{1}{2}$*.*

We are now ready to present a proof of Theorem 6.5.

*Proof of Theorem 6.5.* The main idea is that, given oracle access to the set of random strings, Theorem 6.7 tells us that there is a succinct witness for any exponential-time computation. We abbreviate $R_{\mathrm{K}_U}$ as $R$ in this proof. We only give a detailed proof for $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^R$, since it is straightforward to extend the proof.

First, we present a proof that $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{EXP}^R$. Let $V$ be an exponential-time verifier for $L \in \Sigma_2^{\mathsf{EXP}}$, and $c$ be a constant such that for every input $x$ of length $n$, it holds that $x \in L$ if and only if there exists a certificate $y$ of length $2^{n^c}$ such that $V(x, y, z)$ accepts for all $z$ of length $2^{n^c}$; $V$ runs in time $2^{n^{O(1)}}$. We regard the computation as a game between the first player $A$ and the second player $B$.

Here is our exponential-time algorithm solving $L$ with oracle access to $R$: Let $s_A(n)$ and $s_B(n)$ be some polynomials specified later. Given input $x$ of length $n$, the algorithm accepts if and only if there exists an oracle circuit $A$ of size $s_A(n)$ such that $V(x, \mathtt{tt}(A^R), \mathtt{tt}(B^R))$ accepts for all oracle circuits $B$ of size $s_B(n)$, where $\mathtt{tt}(A^R)$ denotes the truth table of the function computed by $A^R$; the algorithm checks this condition by an exhaustive search. Since there are at most exponentially many circuits of polynomial size, this algorithm runs in exponential time.

We claim the correctness of the algorithm. Fix any input $x \in L$ of length $n$. In this case, the correctness readily follows from the fact that there exists a succinct witness under the oracle $R$: Indeed, let $y_x$ be the lexicographically minimum certificate for $x \in L$. Since each bit of $y_x$ is decidable (in the sense that the language $\{ (x, i) \mid \text{the } i\text{th bit of } y_x \text{ is } 1 \}$ is decidable), by Theorem 6.7, there exists an oracle circuit $A$ of size $s_A(n) := \mathsf{poly}(n, \log |y_x|) = \mathsf{poly}(n, n^c)$ such that $\mathtt{tt}(A^R) = y_x$. Thus our $\mathsf{EXP}^R$ algorithm accepts no matter how the adversarial circuit $B$ is chosen.

Now fix any input $x \notin L$ of length $n$. This case requires a more delicate argument. Here we need to claim that, for every circuit $A$ of size $s_A$, there exists

a circuit $B$ that encodes a strategy that beats the strategy of $A^R$. The point is that, given any circuit $A$, the lexicographically first strategy against the strategy of $A^R$ is computable with oracle access to $\mathsf{HALT}$. Indeed, let $z_{x,A}$ denote the lexicographically first string such that $V(x, \mathtt{tt}(A^R), z)$ rejects. Consider the language $L' := \{\, (x, A, i) \mid \text{the } i\text{th bit of } z_{x,A} \text{ is } 1 \,\}$. Since $R$ is reducible to $\mathsf{HALT}$, the language $L'$ is computable with oracle access to $\mathsf{HALT}$. By Lemma 6.6, $L' \in \mathsf{P}^{\mathsf{HALT}}$; thus by Theorem 6.7, we obtain $L' \in \mathsf{P}^R/\mathsf{poly}$. This means that for every circuit $A$ there exists a circuit $B_A$ of size $\mathsf{poly}(n, |A|, \log |z_{x,A}|) = \mathsf{poly}(n, s_A(n), n^c)$ such that $\mathtt{tt}(B_A^R) = z_{x,A}$. Thus our algorithm rejects.

In order to extend the proof above to $\Sigma_{2k}^{\mathsf{EXP}}$ for every constant $k$, we modify the $\mathsf{EXP}^R$ algorithm so that it checks whether, given input $x$ of length $n$, $\exists$ a circuit $C_1$ of size $s_1(n)$, $\forall$ a circuit $C_2$ of size $s_2(n)$, $\cdots$, $\forall$ a circuit $C_{2k}$ of size $s_{2k}(n)$ such that a verifier $V(x, \mathtt{tt}(C_1^R), \cdots, \mathtt{tt}(C_{2k}^R))$ accepts, where $s_1(n), \cdots, s_{2k}(n)$ are some appropriately chosen polynomials.

We now explain how to reduce the complexity of the $\mathsf{EXP}$ reduction to $\mathsf{PH}$. For simplicity, we again focus on a proof of $\Sigma_2^{\mathsf{EXP}} \subseteq \mathsf{PH}^R$. Note that, in the proof above, the bottleneck of the computation is the evaluation of $V(x, \mathtt{tt}(A^R), \mathtt{tt}(B^R))$, where $V$ is an exponential-time verifier. We replace $V$ with the randomized polynomial-time verifier of Theorem 6.8; then we obtain the following $\Sigma_2^R$ algorithm: Existentially guess a circuit $A$ of size at most $s_A(n)$, and universally guess a circuit $B$ of size at most $s_B(n)$ as well as a coin flip for $V$, Then accept if and only if $V^{A,B}(x)$ accepts on the guessed coin flip sequence. □

### 6.2.2 Proof of Theorem 6.4

A similar technique gives the exact characterization of $\mathsf{S}_2^{\mathrm{exp}}$ in terms of black-box reductions to a dense subset of Kolmogorov-random strings.

**Theorem 6.9** (The formal version of Theorem 6.4)**.** *Let $s\colon \mathbb{N} \to \mathbb{N}$ be a function such that $\ell^\epsilon \le s(\ell) \le \ell - 2$ for some constant $\epsilon > 0$ and every large $\ell$. Let $\gamma\colon \mathbb{N} \to [0, \frac{1}{2}]$ be a function such that $\gamma(\ell) \ge 1/\ell^c$ for some constant $c > 0$ and every large $\ell$. Fix any universal Turing machine $U$. Let $G_\ell\colon \{0,1\}^{s(\ell)} \to \{0,1\}^\ell$ be a function such that $G_\ell(d) = U(d)$ for every $d \in \{0,1\}^{s(\ell)}$ such that $|U(d)| = \ell$. Then we have*

$$\bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{EXP}^{R^{\le \mathsf{poly}}} \quad = \quad \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPEXP}^{R^{\le \mathsf{poly}}} \quad = \quad \mathsf{S}_2^{\mathrm{exp}}.$$

We start with the upper bound of $\mathsf{S}_2^{\mathrm{exp}}$.

**Claim 6.10.**

$$\bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPEXP}^{R^{\le \mathsf{poly}}} \quad \subseteq \quad \mathsf{S}_2^{\mathrm{exp}}.$$

*Proof.* Let $L \in \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{BPEXP}^{R^{\le \mathsf{poly}}}$. We first note that, as in Proposition 5.6, the order of quantifiers can be swapped; indeed, the proof of Proposition 5.6 does not rely on any specific property of $\mathsf{BPP}_{\|}$ reductions; hence, the same proof works for other notions of reduction. Thus, there exists some randomized $t(n)$-time black-box reduction $M$ from a language $L$ to any $\gamma$-avoiding oracle for $G$ such that the length of any query that $M$ makes on input of length $n$ is at most $\log t(n)$, for some $t(n) = 2^{n^{O(1)}}$. Let $L' := \{\, x01^{t(n)} \mid x \in L \,\}$ be a padded version of $L$. Applying Theorem 5.7 to $L'$, we obtain $L' \in \mathsf{S}_2^{\mathrm{p}}$, from which it follows that $L \in \mathsf{S}_2^{\mathrm{exp}}$. □

We now turn to the lower bound, which concludes the proof of Theorem 6.9.

**Claim 6.11.**

$$\mathsf{S}_2^{\exp} \subseteq \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{EXP}^{R^{\leq \mathsf{poly}}}.$$

*Proof.* First, observe that Theorem 6.7 can be generalized to any dense subset $R$ of Kolmogorov-random strings. Indeed, the proof of [ABK+06b] only exploits the property that the set of Kolmogorov-random strings can be used as a distinguisher for a computable hitting set generator. Thus we have $\mathsf{HALT} \in \mathsf{P}^R/\mathsf{poly}$. In particular, relative to the oracle $R$, there exists a succinct witness for $\mathsf{S}_2^{\exp}$.

Let $L \in \mathsf{S}_2^{\exp}$ and $V$ be an exponential-time verifier associated with $L$ running in time $2^{n^k}$ for some constant $k$: That is, for every input $x$ of length $n$,

1. if $x \in L$ then $\exists y \in \{0,1\}^{2^{n^k}}, \forall z \in \{0,1\}^{2^{n^k}}, V(x,y,z) = 1$, and
2. if $x \notin L$ then $\exists z \in \{0,1\}^{2^{n^k}}, \forall y \in \{0,1\}^{2^{n^k}}, V(x,y,z) = 0$.

Our $\mathsf{EXP}^{R^{\leq \mathsf{poly}}}$ algorithm is as follows: Instead of doubly exponentially many certificates $y, z \in \{0,1\}^{2^{n^k}}$, we exhaustively search all possible oracle circuits $Y, Z$ of size at most $\mathsf{poly}(n)$ that take $n^k$ inputs, check the condition that $\exists Y, \forall Z, V(x, \mathtt{tt}(Y^R), \mathtt{tt}(Z^R)) = 1$, and accept if and only if this condition is satisfied. The correctness follows from the fact that each bit of the lexicographically first witness can be computed by a polynomial-size circuit with oracle access to $R$. □

We mention that the reducibility notion of Theorem 6.9 can be significantly improved by using some efficient proof system:

**Theorem 6.12.** *For any $G$ and $\gamma$ satisfying the same condition with Theorem 6.9, we have*

$$\mathsf{EXP}^{\mathsf{NP}} \quad \subseteq \quad \bigcap_{R\colon \gamma\text{-avoids } G} \mathsf{S}_2^R \quad \subseteq \quad \mathsf{S}_2^{\exp}.$$

The proof follows from the following two lemmas:

**Lemma 6.13** ([Hir15])**.** *For any $\mathsf{EXP}^{\mathsf{NP}}$-complete language $L$, there exists a* selector *for $L$. That is, there exists a randomized polynomial-time oracle machine $S$ such that, for any input $x \in \{0,1\}^*$ and oracles $A_0, A_1 \subseteq \{0,1\}^*$, if $L \in \{A_0, A_1\}$ then $\Pr_S\left[S^{A_0,A_1}(x) = L(x)\right] \geq \frac{2}{3}$.*

The next lemma generalizes a lowness property of $\mathsf{S}_2^{\mathrm{p}}$ proved by Cai, Chakaravarthy, Hemaspaandra, and Ogihara [CCHO05].

**Lemma 6.14.** *Let $L$ be a language with a selector and $R$ be any oracle. If $L \in \mathsf{P}^R/\mathsf{poly}$ then $\mathsf{S}_2^L \subseteq \mathsf{S}_2^R$.*

*Proof.* The idea is as follows: We request two competing prover of $\mathsf{S}_2^R$ to output a circuit that computes $L$ relative to $R$, and then we identify the honest oracle by using the selector for $L$.

Suppose that $A \in \mathsf{S}_2^L$, and $M$ be an $\mathsf{S}_2^L$-machine that witnesses $A \in \mathsf{S}_2^L$: for any input $x \in \{0,1\}^n$, we have

$$\begin{aligned} &\exists y, \forall z, \quad M^L(x,y,z) = L(x), \\ &\exists z, \forall y, \quad M^L(x,y,z) = L(x). \end{aligned}$$

Let $c$ be a large constant so that $M$ runs in time $n^c$. Let $S$ be a selector for $L$ that runs in time $n^d$, for some constant $d$.

Now we describe an $S_2^R$-machine that computes $A$: We first show how to compute $L$, given instance $q$ of length at most $n^c$. We request competing provers to give oracle circuits $C_0^R, C_1^R$ that computes $L$ on all inputs of length $\leq n^{cd}$. Then, we run the selector $S$ relative to these circuits. Since the selector makes a query of length at most $n^{cd}$, one of these circuit is an honest oracle. Thus, with high probability, the selector outputs $L(q)$ correctly.

Given input $x \in \{0,1\}^n$, we request two competing provers to output $y, z$. Then compute and output $M^L(x, y, z)$, where the queries $q$ to $L$ can be answered by using the algorithm described above. This algorithm yields an $\mathsf{S}_2 \cdot \mathsf{BP} \cdot \mathsf{P}^R$ algorithm, but since $\mathsf{BPP}$ is low for $\mathsf{S}_2$ [RS98], we obtain $A \in \mathsf{S}_2^R$. □

*Proof of Theorem 6.12.* Under any dense subset $R$ of Kolmogorov-random strings, we have $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^R/\mathsf{poly}$ (by Theorem 6.7). Thus by taking any $\mathsf{EXP}^{\mathsf{NP}}$-complete problem $L$, we obtain $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^L \subseteq \mathsf{S}_2^R$ by combining Lemma 6.13 and 6.14. □

We conclude this chapter by showing that in the case of reductions to the set of random strings, the $\mathsf{S}_2^{\mathrm{p}}$ reductions of Theorem 6.12 can be derandomized to obtain $\mathsf{P}^{\mathsf{NP}}$ reductions.

**Theorem 6.15.** $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$.

*Proof.* By Theorem 6.12, we immediately obtain $\mathsf{EXP}^{\mathsf{NP}} \subseteq \mathsf{S}_2^{R_{\mathrm{K}_U}}$. By the relativized version of Cai's result [Cai07], we have $\mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}} \subseteq \mathsf{S}_2^{R_{\mathrm{K}_U}} \subseteq \mathsf{ZPP}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$; thus it remains to derandomize the computation of $\mathsf{ZPP}$ under an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle. One can find the lexicographically first Kolmogorov-random string by a $\mathsf{P}^{\mathsf{NP}^{R_{\mathrm{K}_U}}}$ algorithm. By Lemma 6.6 and Theorem 6.7, the circuit complexity relative to an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle of any Kolmogorov-random string of length $n$ is at least $n^{\Omega(1)}$. Thus by using a Kolmogorov-random string as a hard function of the Impagliazzo-Wigderson pseudorandom generator [IW97], one can derandomize the computation of $\mathsf{ZPP}$ under an $\mathsf{NP}^{R_{\mathrm{K}_U}}$ oracle. □

# Chapter 7

# Hardness of MCSP Implies Circuit Lower Bounds

So far we have seen that various hardness results of MCSP under average-case hardness assumptions. A natural question is what prevents us from proving NP-hardness of MCSP.

In this chapter, we show that MCSP is intimately related to circuit lower bounds, of which we have a poor understanding. In particular, under *deterministic* reductions, MCSP cannot be shown to be NP-complete unless important open questions are solved simultaneously.

## 7.1 Overview

There have been a line of work explaining why it is difficult to establish NP-hardness of MCSP under *deterministic* reductions (*e.g.* [KC00, MW17, AHK17, HP15]). For example, Murray and Williams [MW17] showed that if MCSP is NP-hard under polynomial-time many-one reductions, then ZPP $\neq$ EXP, thereby showing that NP-hardness of MCSP is at least as hard as proving ZPP $\neq$ EXP. However, no previous work was able to obtain a similar consequence for polynomial-time Turing reductions. We extend previous results to the case of polynomial-time nonadaptive reductions and polynomial-time Turing reductions.

In fact, we prove several unconditional separations about deterministic polynomial-time machines with oracle access to MCSP. Specifically, in the case of nonadaptive reductions, we show that $\mathsf{P}_{\parallel}^{\mathrm{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$; here $\mathsf{P}_{\parallel}$ denotes the class of languages reducible to MCSP via a polynomial-time nonadaptive reduction. In the case of adaptive reductions, we show that $\mathsf{P}^{\mathrm{Gap}_\epsilon \mathrm{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$ for some sufficiently small constant $\epsilon > 0$. As a consequence, we show that NP-hardness (or even ZPP-hardness) of MCSP under nonadaptive deterministic reductions implies EXP $\neq$ ZPP, the latter of which is a notorious open question in complexity theory. Moreover, we show that NP-hardness of $\mathrm{Gap}_\epsilon$MCSP under *adaptive* deterministic reductions for all constant $\epsilon > 0$ implies NEXP $\not\subseteq$ P/poly. These results explain why it is difficult to prove hardness of MCSP under deterministic reductions.

Our idea is based on the firm links between circuit complexity and resource-bounded Kolmogorov complexity as explored by [ABK$^+$06b, AKRR11]. Specifically, under the assumption that EXP $\subseteq$ P/poly, it is known that circuit complexity and Levin's Kt-complexity are polynomially related to each other. In this sense, we can translate any property of (an approximation version of) MKtP into that of MCSP. On the other hand, one can observe that $\mathsf{P}_{\parallel}^{\mathrm{MKtP}} \subseteq \mathsf{DTIME}(2^{2n}) \subsetneq \mathsf{EXP}$; this is because the Kt-complexity of any query that a polynomial-time ma-

chine can make is bounded above by $n + O(\log n)$, and thus the Kt-complexity of such a query can be determined by an exhaustive search in time $2^{n+O(\log n)}$. Now we can translate this property of MKtP into that of MCSP and obtain $\mathsf{P}_{\|}^{\mathrm{MCSP}} \neq \mathsf{EXP}$, assuming that $\mathsf{EXP} \subseteq \mathsf{P/poly}$. As a consequence, we obtain $\mathsf{P}_{\|}^{\mathrm{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$ *unconditionally* because otherwise we have $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

Now we would like to extend the argument above into the case of polynomial-time Turing reductions. Unfortunately, we do not know how to prove $\mathsf{EXP} \neq \mathsf{P}^{\mathrm{MKtP}}$ (and this is an open problem since [ABK+06b]). Nevertheless, we can still prove that a promise problem (denoted by $\mathrm{Gap}_g\mathrm{MKtP}$) of approximating Kt within an additive error $\omega(\log n)$ is not $\mathsf{EXP}$-hard under polynomial-time Turing reductions, that is, $\mathsf{P}^{\mathrm{Gap}_g\mathrm{MKtP}} \neq \mathsf{EXP}$. We can translate the property of Kt-complexity into that of MCSP, under the assumption that $\mathsf{EXP} \subseteq \mathsf{P/poly}$. While the quality of approximation becomes worse, we are able to show that $\mathsf{P}^{\mathrm{Gap}_\epsilon\mathrm{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$ unconditionally.

We will also show that $\mathsf{NP}$-hardness of MCSP implies a *strongly exponential* circuit lower bound (or else a fast simulation of nondeterministic machines). In particular, the result indicates that it is already a challenging open question to prove $\mathsf{NP}$-hardness of MCSP for depth-3 $\mathsf{AC}^0$ circuits, for which no strongly exponential circuit lower bound is known.

Our proof idea for this is based on the original proof idea of Kabanets and Cai [KC00], and its later exposition of Hitchcock and Pavan [HP15]: If there is a many-one reduction $R$ from $\mathsf{SAT}$ to MCSP, then by running the reduction $R$ on an unsatisfiable formula, we obtain a NO instance of MCSP, which is a truth table of high circuit complexity. Thus we should be able to obtain some circuit lower bound.

## 7.2 The Case of Nonadaptive Reductions

Our proof is based on the following relationship between Kt-complexity and circuit complexity:

**Lemma 7.1** (Allender, Koucký, Ronneburger and Roy [AKRR11])**.** $\mathsf{EXP} \subseteq \mathsf{P/poly}$ *if and only if there exists a polynomial* $p$ *such that* $\mathsf{size}(x) \leq p(\mathrm{Kt}(x), \log|x|)$ *for every* $x \in \{0,1\}^*$.

We will first show $\mathsf{P}_{\|}^{\mathrm{MKtP}} \neq \mathsf{EXP}$, and then, based on Lemma 7.1, translate the property of Kt into that of MCSP, assuming $\mathsf{EXP} \subseteq \mathsf{P/poly}$.

**Proposition 7.2** (folklore)**.** $\mathsf{P}_{\|}^{\mathrm{MKtP}} \subseteq \mathsf{DTIME}(2^{2n}) \subsetneq \mathsf{EXP}$.

*Proof.* The last "$\subsetneq$" follows from the time hierarchy theorem [HS65].

Let $M$ be any $\mathsf{P}_{\|}^{\mathrm{MKtP}}$ machine. Given input $x \in \{0,1\}^*$ of length $n$, let $Q(x)$ be the set of queries (without including size parameter $s$) that $M$ makes. Since $M$ is a nonadaptive polynomial-time oracle machine, $Q(x)$ can be computed in polynomial time. Therefore, any query $q \in Q(x)$ can be described by the input $x$ and an index $i \in [n^{O(1)}]$ in polynomial time; hence, $\mathrm{Kt}(q) \leq n + O(\log n)$. Let $l(n)$ denote this upper bound.

Given the fact that $\mathrm{Kt}(q) \leq l(n)$, we can compute $\mathrm{Kt}(q)$ by an exhaustive search in time $2^{n+O(\log n)}$. Specifically, for each $d \in \{0,1\}^*$ of length at most $l(n)$, we run the universal Turing machine $U$ on input $d$ for $2^{l(n)-|d|}$ steps; this takes overall $2^{l(n)}n^{O(1)}$ time. Thus, by answering $M$'s queries by the exhaustive search, we can compute $M$'s output in time $2^{n+O(\log n)} \leq 2^{2n}$. □

Under the assumption that $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we can translate the property of MKtP into that of MCSP:

**Theorem 7.3.** *If* $\mathsf{EXP} \subseteq \mathsf{P/poly}$ *then* $\mathsf{P}_{\|}^{\mathrm{MCSP}} \subseteq \mathsf{DTIME}(2^{n^c}) \neq \mathsf{EXP}$ *for some constant* $c > 0$.

*Proof Sketch.* By Lemma 7.1, there exists a constant $c$ such that $\mathsf{size}(x) \leq (\mathrm{Kt}(x) + \log|x|)^c$ for all large $|x|$. Let $(q, s)$ be any query of a $\mathsf{P}_{\|}^{\mathrm{MCSP}}$ machine. Since $\mathrm{Kt}(q)$ is at most $n + O(\log n)$, the circuit complexity $\mathsf{size}(q)$ of $q$ is bounded above by $(\mathrm{Kt}(q) + \log n)^c \leq (2n)^c$. Thus, the circuit complexity of all the queries can be computed by an exhaustive search in time $2^{O(n^c)\log O(n^c)}$. □

This theorem allows us to obtain a nontrivial separation of $\mathsf{P}_{\|}^{\mathrm{MCSP}} \cap \mathsf{P/poly}$ from $\mathsf{EXP}$:

**Corollary 7.4.** $\mathsf{P}_{\|}^{\mathrm{MCSP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$.

*Proof.* Assume, by way of contradiction, that $\mathsf{P}_{\|}^{\mathrm{MCSP}} \cap \mathsf{P/poly} = \mathsf{EXP}$. In particular, $\mathsf{EXP} \subseteq \mathsf{P/poly}$. Thus, by Theorem 7.3, we obtain $\mathsf{P}_{\|}^{\mathrm{MCSP}} \neq \mathsf{EXP}$, which is a contradiction to the assumption. □

This result exhibits a singular property of MCSP. In particular, reducing any language $L$ to MCSP via a polynomial-time nonadaptive reduction implies a separation of $\mathsf{P}_{\|}^{L} \cap \mathsf{P/poly}$ from $\mathsf{EXP}$.

**Corollary 7.5.** *If* $L \leq_{\|}^{\mathsf{P}} \mathrm{MCSP}$, *then* $\mathsf{P}_{\|}^{L} \cap \mathsf{P/poly} \neq \mathsf{EXP}$.

*Proof.* The hypothesis implies that $\mathsf{P}_{\|}^{L} \subseteq \mathsf{P}_{\|}^{\mathrm{MCSP}}$, and by the previous corollary it holds that $\mathsf{EXP} \not\subseteq \mathsf{P}_{\|}^{\mathrm{MCSP}} \cap \mathsf{P/poly}$, from which the result follows. □

In particular, if MCSP is $\mathsf{ZPP}$-hard under polynomial-time nonadaptive reductions, then $\mathsf{ZPP} \neq \mathsf{EXP}$. Similarly, if MCSP is $\mathsf{NP}$-hard under the same reducibility notion, then $\mathsf{P}_{\|}^{\mathsf{NP}} \cap \mathsf{P/poly} \neq \mathsf{EXP}$.

## 7.3 The Case of Adaptive Reductions

Now we turn to the case of polynomial-time Turing reductions. We first show that approximating Kt-complexity within an additive error $\omega(\log n)$ is not $\mathsf{EXP}$-complete under polynomial-time Turing reductions. We denote such a promise problem by $\mathrm{Gap}_g\mathrm{MKtP}$:

**Definition 7.6.** *For a function* $g\colon \mathbb{N} \to \mathbb{N}$, *define a promise problem* $\mathrm{Gap}_g\mathrm{MKtP} := (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *by*

$$\Pi_{\mathrm{YES}} := \{\, (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathrm{Kt}(x) \leq s \,\},$$
$$\Pi_{\mathrm{NO}} := \{\, (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathrm{Kt}(x) > s + g(|x|) \,\}.$$

**Theorem 7.7.** *For any nondecreasing function* $g(n) = \omega(\log n)$, *it holds that* $\mathsf{P}^{\mathrm{Gap}_g\mathrm{MKtP}} \neq \mathsf{EXP}$.

We mention that the proof is reminiscent of a simplified proof in [ABK+06b, Corollary 40] showing that resource-bounded Kolmogorov complexity $\mathrm{K}_{t(\text{-})}$ for a fixed exponential time $t(n) \geq 2^{n^2}$ is not $\mathsf{EXP}$-hard (originally proved by Buhrman and Mayordomo [BM97]).

*Proof.* It is sufficient to prove that every unary language in $\mathsf{P}^{\mathrm{Gap}_g\mathrm{MKtP}}$ can be solved in a fixed exponential time. Indeed, by the time hierarchy theorem, there exists a unary language in $\mathsf{EXP}$ that requires time complexity larger than any fixed exponential time, which implies that $\mathsf{P}^{\mathrm{Gap}_g\mathrm{MKtP}} \neq \mathsf{EXP}$.

We first note that $\mathrm{Kt}(x) \leq |x| + O(\log |x|)$ for any $x \in \{0,1\}^*$, since every string can be described by itself in polynomial time. Let $l(n)$ be such a (nondecreasing) upper bound (*i.e.* $l(n) = n + O(\log n)$).

Let $L \subseteq \{0\}^*$ be an arbitrary unary language in $\mathsf{P}^{\mathrm{Gap}_g\mathrm{MKtP}}$, and $M$ be a polynomial-time machine that witnesses $L \in \mathsf{P}^{\mathrm{Gap}_g\mathrm{MKtP}}$.

The proof idea is as follows: We would like to simulate $M$ on input $0^n$ without oracle access to $\mathrm{Gap}_g\mathrm{MKtP}$ in time $2^{2n} \ll 2^{n^{O(1)}}$. To this end, we try to answer $M$'s query $q$ by exhaustively searching up to Kt-complexity $l(n)$. While we cannot obtain the correct value $\mathrm{Kt}(q)$ for a query $q$ such that $\mathrm{Kt}(q) > l(n)$, we guess the value $\mathrm{Kt}(q)$ to be $l(n)$. Then, we will argue that each query $q$ can be computed efficiently and hence $\mathrm{Kt}(q)$ is relatively small; therefore, the guessed value of Kt-complexity gives a good approximation. A formal proof follows.

We define a machine $M_0$ that simulates $M$ on input $0^n$ (without oracle access to $\mathrm{Gap}_g\mathrm{MKtP}$): On input $0^n$, $M_0$ simulates $M$ on the same input, and accepts if and only if $M$ accepts. If the machine $M$ makes a query $(q, s) \in \{0,1\}^* \times \mathbb{N}$ to a $\mathrm{Gap}_g\mathrm{MKtP}$ oracle, then we perform an exhaustive search up to Kt-complexity $l(n)$, which allows us to compute $\sigma_n(q) := \min\{\mathrm{Kt}(q), l(n)\}$. (Namely, for each $d \in \{0,1\}^*$ of length at most $l(n)$, run the universal Turing machine $U$ on input $d$ for time $2^{l(n)-|d|}$, which takes overall $2^{l(n)}n^{O(1)}$ time.) We answer "Yes" to the query $q$ if and only if $\sigma_n(q) \leq s$. The machine $M_0$ runs in time $2^{l(n)}n^{O(1)} \leq 2^{2n}$ (*i.e.* a fixed exponential time). Hence, it remains to prove that, for each $n \in \mathbb{N}$, there exists an oracle $A$ that satisfies the promise of $\mathrm{Gap}_g\mathrm{MKtP}$ such that $M_0(0^n) = M^A(0^n)$, which in particular implies that $M_0(0^n) = L(0^n)$.

A crucial observation here is that each query that $M$ makes on the computation path simulated by $M_0$ can be described succinctly in terms of Kt-complexity: Specifically, fix an input $0^n$ and define the set $Q_n = \{(q_1, s_1), \cdots, (q_m, s_m)\}$ of queries that $M$ makes on the computation path simulated by $M_0$, where $m = n^{O(1)}$ is the number of the queries. Then, the $i$th query $(q_i, s_i)$ can be described by $n$ and an index $i \in [m]$ in time $2^{l(n)}n^{O(1)}$. Therefore, it holds that $\mathrm{Kt}(q_i) \leq O(\log n) + \log 2^{l(n)}n^{O(1)} = l(n) + O(\log n)$. By the assumption, we have $O(\log n) \leq g(n)$ for all large $n$; hence, $\mathrm{Kt}(q_i) \leq l(n) + g(n)$. This means that the difference between $\mathrm{Kt}(q_i)$ and the threshold $l(n)$ up to which we performed an exhaustive search is at most $g(n)$.

Now, for each $n \in \mathbb{N}$, define an oracle $A$ as follows: $(q, s) \in A$ if and only if $\sigma_n(q) \leq s$ for any $(q, s) \in Q_n$, and $(q, s) \in A$ if and only if $\mathrm{Kt}(q) \leq s$ for any $(q, s) \notin Q_n$. (Here, $\sigma_n(q)$ denotes $\min\{\mathrm{Kt}(q), l(n)\}$.) By this definition, it holds that $M^A(0^n) = M_0(0^n)$; thus all that remains is to show that $A$ satisfies the promise of $\mathrm{Gap}_g\mathrm{MKtP}$ (which implies that $M^A(0^n) = L(0^n)$).

Namely, for all $(q, s) \in Q_n$, we would like to claim that $(q, s) \in A$ holds if $(q, s)$ is an YES instance of $\mathrm{Gap}_g\mathrm{MKtP}$ (*i.e.* $\mathrm{Kt}(q) \leq s$), and that $(q, s) \notin A$ holds if $(q, s)$ is a NO instance of $\mathrm{Gap}_g\mathrm{MKtP}$ (*i.e.* $\mathrm{Kt}(q) \geq s + g(|q|)$). Note that if $\mathrm{Kt}(q) \leq l(n)$ then $\sigma_n(q) = \mathrm{Kt}(q)$; hence in this case, the claim is obviously satisfied. In what follows, we may assume that $\mathrm{Kt}(q) > l(n)$ (and thus $\sigma_n(q) = l(n)$). In particular, this implies that $n \leq |q|$: indeed, by the definition of $l(n)$, we have $\mathrm{Kt}(q) \leq l(|q|)$, which implies $l(n) < \mathrm{Kt}(q) \leq l(|q|)$; hence, $n \leq |q|$ follows. Therefore, $\mathrm{Kt}(q) \leq l(n) + g(n) \leq l(n) + g(|q|)$. Now assume that $\mathrm{Kt}(q) > s + g(|q|)$ (*i.e.* $(q, s)$ is a NO instance). This implies that $\sigma_n(q) = l(n) \geq \mathrm{Kt}(q) - g(|q|) > s$,

and hence $(q, s) \notin A$ as desired. On the other hand, if $\mathrm{Kt}(q) \leq s$ (*i.e.* $(q, s)$ is an YES instance), then we have $\sigma_n(q) \leq \mathrm{Kt}(q) \leq s$, and hence $(q, s) \in A$. □

Next, assuming that $\mathsf{EXP} \subseteq \mathsf{P/poly}$, we translate the property of Kt-complexity into that of MCSP. However, since these two measures are just polynomially related, the narrow gap of Kt does not seem to be translated into a narrow gap of MCSP. We can still prove analogous results for a promise problem $\mathrm{Gap}^k\mathrm{MCSP}$ asking for approximating the logarithm of circuit complexity within a factor of $k$. More formally:

**Definition 7.8.** *For a constant $k \geq 1$, define a promise problem* $\mathrm{Gap}^k\mathrm{MCSP} := (\Pi_{\mathrm{YES}}, \Pi_{\mathrm{NO}})$ *by*

$$\Pi_{\mathrm{YES}} := \{\, (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathsf{size}(x) \leq s \,\},$$
$$\Pi_{\mathrm{NO}} := \{\, (x, s) \in \{0,1\}^* \times \mathbb{N} \mid \mathsf{size}(x) > s^k \,\}.$$

We note that this problem is harder than $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, and thus our results will be stronger.

**Fact 7.9.** *For any constants $k \in \mathbb{N}, \epsilon > 0$ such that $k\epsilon \leq 1$,*

$$\mathrm{Gap}_\epsilon\mathrm{MCSP} \leq^{\mathsf{P}}_{\mathrm{m}} \mathrm{Gap}^k\mathrm{MCSP}.$$

*Proof.* Take any input $(\mathtt{tt}(f), s)$ where $f\colon \{0,1\}^n \to \{0,1\}$. Recall that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is a promise problem of approximating circuit complexity within a factor of $2^{(1-\epsilon)n}$. Therefore, if $s \geq 2^{\epsilon n}$, then we are allowed to answer "YES" as an answer to $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ because $s \cdot 2^{(1-\epsilon)n} \geq 2^n$. We thus map the input to a trivial YES instance (say, $(\mathtt{tt}(f), 2^n)$). In the case when $s < 2^{\epsilon n}$, we just map the input to the same input $(\mathtt{tt}(f), s)$.

We claim the correctness of the reduction in the case of $s < 2^{\epsilon n}$. It is obvious that an YES instance of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is mapped to an YES instance of $\mathrm{Gap}^k\mathrm{MCSP}$ by the definitions of these promise problems. For any NO instance of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, we have $\mathsf{size}(f) > s \cdot 2^{(1-\epsilon)n} \geq s \cdot 2^{\epsilon n(k-1)} \geq s^k$, and hence $(\mathtt{tt}(f), s)$ is a NO instance of $\mathrm{Gap}^k\mathrm{MCSP}$. □

Now we apply the same proof idea with Theorem 7.7 to $\mathrm{Gap}^k\mathrm{MCSP}$. In fact, thanks to the fact that the gap between $\Pi_Y$ and $\Pi_N$ is wide, we can prove a stronger consequence:

**Theorem 7.10.** *If $\mathsf{EXP} \subseteq \mathsf{P/poly}$, then for any $\epsilon > 0$, there exists a constant $k \geq 1$ such that $\mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \subseteq \mathsf{DTIME}(2^{n^\epsilon})$. In particular, $\mathsf{EXP} \neq \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \cap \mathsf{P/poly}$ for some $k$.*

We note that by Fact 7.9, we also have $\mathsf{P}^{\mathrm{Gap}_{1/k}\mathrm{MCSP}} \subseteq \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \subseteq \mathsf{DTIME}(2^{n^\epsilon})$.

*Proof.* The proof idea is exactly the same with that of Theorem 7.7: We first simulate a $\mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}}$ machine by answering its query $T$ by an exhaustive search up to circuit complexity $l(n)$ for some $l(n)$. Then, since any query $q$ can be described succinctly in terms of Kt-complexity, the circuit complexity $\mathsf{size}(q)$ of the query $q$ is also relatively small by Lemma 7.1; hence, the incomplete exhaustive search gives a somewhat good approximation. While the theorem can be proved based on Lemma 7.1, we incorporate a proof of Lemma 7.1 and give an entire proof below for completeness.

Let us define an $\mathsf{EXP}$-complete language $B \subseteq \{0,1\}^*$ as all the tuples $\langle Q, x, t\rangle$ such that the Turing machine $Q$ accepts $x$ in time $t$. Since $B \in \mathsf{EXP} \subseteq \mathsf{P/poly}$, there exist some constant $k_0 \in \mathbb{N}$ and some family of circuits $\{C_m\}_{m\in\mathbb{N}}$ of size at most $m^{k_0}$ that computes $B$ on input length $m$.

Fix a small constant $\epsilon > 0$. Define $k := (k_0+1)/\epsilon$. Let $L \in \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}}$ and $M$ be a polynomial-time oracle machine that witnesses $L \in \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}}$.

Define $l(n) := n^\epsilon$. As in the proof of Theorem 7.7, we define a machine $M_0$ that simulates $M$ (without oracle access to $\mathrm{Gap}^k\mathrm{MCSP}$) as follows: $M_0$ takes input $x \in \{0,1\}^*$ of length $n$, simulates $M$ on input $x$, and accepts if and only if $M$ accepts. If $M$ makes a query $(q, s)$, then answer to the query by an exhaustive search up to circuit size $l(n)$. (Specifically, compute $\sigma_x(q) := \min\{\mathsf{size}(q), l(n)\}$ and answer "Yes" if and only if $\sigma_x(q) \le s$.) The machine $M_0$ runs in time $2^{l(n)}n^{O(1)} \le 2^{n^{2\epsilon}}$ for all large $n$.

Fix input $x \in \{0,1\}^*$ of length $n$. Let $Q_x = \{(q_1, s_1), \cdots, (q_{n^{O(1)}}, s_{n^{O(1)}})\}$ be the set of all the queries that $M$ makes on the computation path simulated by $M_0$. We claim that for each $(q_i, s_i) \in Q_x$, the circuit complexity $\mathsf{size}(q_i)$ is relatively small: Indeed, each truth table $q_i$ in $Q_x$ can be computed in time $t(n) := 2^{n^{2\epsilon}}$, by simulating $M$ in the same way with $M_0$. Let $Q$ be the Turing machine that takes as input $x \in \{0,1\}^*$ of length $n$ and indices $i, j \in [n^{O(1)}]$, and outputs $q_{ij}$. By the definition of $B$, it holds that $B(Q, \langle x, i, j\rangle, t(n)) = Q(x, i, j) = q_{ij}$. Also, by the definition of $C_m$, we have $B(Q, \langle x, i, j\rangle, t(n)) = C_m(Q, \langle x, i, j\rangle, t(n))$ for $m = |\langle Q, \langle x, i, j\rangle, t(n)\rangle|$. Note that $m = 4n + O(\log n) + \log t(n) \le 5n$ for all large $n$. Now let us fix $x \in \{0,1\}^n$ and $i \in [n^{O(1)}]$: namely, define $D_{x,i}(j) = C_m(Q, \langle x, i, j\rangle, t(n))$; then, the truth table of $D_{x,i}$ coincides with $q_i$. Therefore,

$$\mathsf{size}(q_i) \le |D_{x,i}| \le |C_m| \le m^{k_0} \le (5n)^{k_0} \le n^{k\epsilon} = l(n)^k$$

for all large $n$. (Here, $|C_m|$ denotes the circuit size of $C_m$.)

Now we claim that $\sigma_x(q_i) = \min\{\mathsf{size}(q_i), l(n)\}$ approximates $\mathsf{size}(q_i)$ for all $(q_i, s_i) \in Q_x$: specifically, we claim that $\sigma_x(q_i) \le \mathsf{size}(q_i) < \sigma_x(q_i)^k$. If $\mathsf{size}(q_i) \le l(n)$, then $\sigma_x(q_i) = \mathsf{size}(q_i)$ and the claim is obvious. Now assume that $\mathsf{size}(q_i) > l(n)$, which implies that $\sigma_x(q_i) = l(n)$. Thus we have $\sigma_x(q_i) = l(n) < \mathsf{size}(q_i) < l(n)^k = \sigma_x(q_i)^k$.

From the inequalities above, for every $x \in \{0,1\}^*$, it is easy to see that there exists an oracle $A$ such that $A$ satisfies the promise of $\mathrm{Gap}^k\mathrm{MCSP}$ and $M_0(x) = M^A(x) = L(x)$. □

**Corollary 7.11.** *If* $\mathsf{NP} \le_{\mathrm{T}}^{\mathsf{P}} \mathrm{Gap}^k\mathrm{MCSP}$ *for all* $k \ge 1$*, then* $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$.

*Proof.* Assume, for the purpose of contradiction, that $\mathsf{NEXP} \subseteq \mathsf{P/poly}$. In particular, $\mathsf{EXP} \subseteq \mathsf{P/poly}$. By Theorem 7.10, we obtain $\bigcap_{k\ge1} \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \subseteq \mathsf{SUBEXP}$. By the assumption, we have $\mathsf{NP} \subseteq \bigcap_{k\ge1} \mathsf{P}^{\mathrm{Gap}^k\mathrm{MCSP}} \subseteq \mathsf{SUBEXP}$. In particular, $\mathsf{CircuitSAT} \in \mathsf{SUBEXP}$. It was shown by Williams [Wil13] that $\mathsf{CircuitSAT} \in \mathsf{SUBEXP}$ implies $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$, which is a contradiction. □

**Remark.** Another interesting consequence of Theorem 7.10 is that if MCSP itself is reducible to $\mathrm{Gap}^k\mathrm{MCSP}$ for all $k \ge 1$ via a polynomial-time Turing reduction, then $\mathsf{P}^{\mathrm{MCSP}} \cap \mathsf{P/poly} \ne \mathsf{EXP}$, which we do not know how to prove. Thus, establishing such "robustness" of MCSP via a polynomial-time Turing reduction is at least as hard as separating $\mathsf{P}^{\mathrm{MCSP}} \cap \mathsf{P/poly}$ from $\mathsf{EXP}$.

## 7.4 Hardness of MCSP and Stronger Circuit Lower Bounds

In the previous two sections, we have focused on polynomial-size circuits, and tried to establish the relationship between hardness of MCSP and polynomial-size circuit lower bounds. However, for restricted class of circuits such as depth-$d$ $\mathsf{AC}^0$ circuits, an exponential lower bound of the form $2^{\Omega(n^{1/(d-1)})}$ is already known [Hås86]. A natural question is whether MCSP for such a restricted circuit class can be proven to be $\mathsf{NP}$-hard. Here we show that it is unlikely under polynomial-time reductions, unless either the lower bound is as large as $2^{\Omega(n)}$, or there is a fast deterministic simulation of $\mathsf{NE} = \mathsf{NTIME}(2^{O(n)})$.

**Theorem 7.12.** *If* MCSP *is* $\mathsf{NP}$*-hard under polynomial-time Turing reductions, then either* $\mathsf{E}^{\mathrm{MCSP}} \not\subseteq \mathsf{SIZE}(2^{\epsilon n})$ *for some constant* $\epsilon > 0$ *or* $\mathsf{NE} \subseteq \mathsf{SUBEE} := \bigcap_{\epsilon>0} \mathsf{DTIME}(2^{2^{\epsilon n}})$.

While we stated it only for general circuit classes, the same results hold for any reasonable circuit class. In particular, we note that it is an open problem whether $\mathsf{E}^{\mathsf{NP}}$ cannot be computed by a depth-3 $\mathsf{AC}^0$ circuit of size $2^{\omega(\sqrt{n})}$; thus proving $\mathsf{NP}$-hardness of MCSP for depth-3 $\mathsf{AC}^0$ circuits is already challenging.

*Proof.* Assuming that $\mathsf{NE} \not\subseteq \mathsf{SUBEE}$ and $\mathsf{NP} \subseteq \mathsf{P}^{\mathrm{MCSP}}$, it suffices to show that $\mathsf{E}^{\mathrm{MCSP}} \not\subseteq \mathsf{SIZE}(2^{\Omega(n)})$. We take a language $L \in \mathsf{NE} \setminus \mathsf{SUBEE}$. We define a padded version of $L$ as $L' := \{\, 1^n \mid n \in L \,\}$, where $n$ is identified with its binary representation; thus $L' \in \mathsf{NP} \setminus \mathsf{SUBEXP}$. In particular, there exists a constant $\epsilon > 0$ such that $L' \in \mathsf{NP} \setminus \mathsf{DTIME}(2^{n^\epsilon})$.

Let $M$ be a polynomial-time oracle machine that witnesses $L' \in \mathsf{P}^{\mathrm{MCSP}}$. We try to simulate $M$ by the following algorithm $M_0$: On input $1^n$, simulate $M$, and if $M$ makes a query $(q, s)$ to MCSP, then we answer the query by exhaustively searching all the circuits of size at most $n^{\epsilon/2}$.

Note that the running time of $M_0$ is at most $2^{O(n^{\epsilon/2} \log n^{\epsilon/2})} \leq 2^{n^\epsilon}$. Since $L' \notin \mathsf{DTIME}(2^{n^\epsilon})$, the simulation of $M$ by $M_0$ must fail infinitely often. Let $I \subseteq \mathbb{N}$ denote the infinite set of integers $n \in \mathbb{N}$ such that $M$ and $M_0$ differ on input $1^n$.

We now define a hard language $H$ in $\mathsf{E}^{\mathrm{MCSP}}$ by the following algorithm: Given as input binary representations of a tuple $(n, i, j) \in \mathbb{N}^3$ where $i, j \in [n^{O(1)}]$, simulate $M^{\mathrm{MCSP}}(1^n)$ and denote by $q_{n,i}$ the $i$th query that $M$ makes to the MCSP oracle. Output the $j$th bit of $q_{n,i}$.

Note that the input length of the algorithm for $H$ is $\ell := O(\log n)$; thus the running time $n^{O(1)}$ is exponential in $O(\log n)$, and hence the language $H$ is indeed in $\mathsf{E}^{\mathrm{MCSP}}$. Moreover, since the simulation of $M$ by $M_0$ fails on input $1^n$ for $n \in I$, there exists some $i_n \in [n^{O(1)}]$ such that $\mathsf{size}(q_{n,i_n}) > n^{\epsilon/2}$. Thus $H(n, i_n, \text{-})$ requires a circuit of size $2^{\Omega(\log n)}$. Therefore, $H$ requires a circuit of size $2^{\Omega(\ell)}$ infinitely often. □

## 7.5 Open Questions

We conclude this chapter by posing two open questions. The results of this section can be interpreted as "proving $\mathsf{NP}$-hardness of MCSP is at least as difficult as open problems such as $\mathsf{ZPP} \neq \mathsf{EXP}$."

**Open Question 7.13.** Is it possible to show that "MCSP is unlikely to be $\mathsf{NP}$-complete"? Specifically, show that $\mathsf{NP}$-hardness of MCSP implies something unlikely to be true.

Note that the results presented this section only imply something that is widely believed to be true, but simply we do not know how to prove (*e.g.* $\mathsf{ZPP} \neq \mathsf{EXP}$). There are two known results of the form of Open Question 7.14: One is the result of Murray and Williams [MW17] showing that MCSP is *provably* not $\mathsf{NP}$-hard under $\mathsf{DTIME}(n^{0.49})$-time reductions. We will present another result in Chapter 8 showing that $\mathsf{NP}$-hardness of MCSP under "oracle-independent" deterministic reductions implies $\mathsf{NP} = \mathsf{P}$.

The other open question is to extend the results of this section to randomized reductions.

**Open Question 7.14.** Show that $\mathsf{NP}$-hardness of MCSP under *randomized reductions* implies some open question in complexity theory (similar to Theorem 7.12).

Intuitively, all the techniques presented in this section exploit the idea that $\mathsf{NP}$-hardness of MCSP means that one can efficiently construct a No instance of MCSP. Under randomized reductions, one cannot make use of such an idea because a random truth table is a No instance of MCSP with high probability. Moreover, as shown in Chapter 3, randomized reductions enable us to show $\mathsf{SZK}$-hardness of MCSP; in contrast, in the case of deterministic reductions, no nontrivial reduction is known unconditionally. In this sense, the results presented in this section highlight that deterministic reductions are insufficient to argue hardness of MCSP. It thus seems to require a significantly new idea to resolve Open Question 7.14. It should be also noted that if $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ then MCSP is $\mathsf{NP}$-hard under $\mathsf{ZPP}$-Turing reductions [ABK$^+$06b, IKV18]; thus if the consequence of Open Question 7.14 were $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$, we would immediately obtain $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$ unconditionally.

In the next chapter, we discuss limits of *randomized reductions* by introducing the new notion called oracle-independent reductions.

# Chapter 8

# Oracle-Independent Reductions

In this chapter, we show fundamental limits of current techniques for showing hardness of MCSP. Specifically, we observe that almost all reduction techniques do not rely on any inherent property of MCSP and instead rely on common properties that $\mathrm{MCSP}^A$ shares for an arbitrary oracle $A$. We thus introduce the notion of *oracle-independent reductions* to MCSP and then give upper bounds on classes of languages that reduces to MCSP via such reductions. We say that a reduction to MCSP is *oracle-independent* if the reduction can be generalized to $\mathrm{MCSP}^A$ for an arbitrary oracle $A$. In other words, the reduction exploits only properties common to $\mathrm{MCSP}^A$ for any oracle $A$ (instead of nonrelativizing properties of MCSP).

Almost all the known reductions to MCSP are oracle-independent. The main technique used in Chapter 3 is the construction from a one-way function to a pseudorandom generator: Specifically, since the output of a pseudorandom (function) generator is efficiently computable, the output regarded as a truth table has significantly low circuit complexity, compared to that of a truth table chosen from a uniform distribution. Thus, MCSP constitutes a statistical test that distinguishes a pseudorandom distribution from a uniform distribution, which enables us to invert a one-way function on average, thanks to [HILL99]. This argument exploits only the fact that MCSP constitutes a statistical test. It is easy to see that an oracle version $\mathrm{MCSP}^A$ can also constitute a statistical test, and hence such reductions are oracle-independent.

Recently, new types of reductions to MCSP that do not rely on inverting a one-way function have been developed by Allender, Grochow, van Melkebeek, Moore, and Morgan [AGvM$^+$18]. Based on new ideas, they showed that a graph isomorphism problem is reducible to MKTP via a randomized reduction with zero-sided error. We will see that their reductions are also oracle-independent.

A high-level reason why these reductions are oracle-independent is as follows: We are prone to rely on the fact that a randomly chosen truth table requires high circuit complexity, because it is in general difficult to obtain a circuit lower bound on an explicit function. The fact that many truth tables require high circuit complexity remains unchanged for any oracle version $\mathrm{MCSP}^A$, and hence a reduction that only exploits this fact (as a circuit lower bound) is inevitably oracle-independent.

We note that there is a trivial exception for oracle-independent reductions: the self-reduction that maps an instance of MCSP to itself. Similarly, the non-black-box reductions of Chapter 4 are not oracle-independent.

## 8.1 Our Results

We provide strong evidence that NP-hardness of MCSP cannot be shown via such oracle-independent reductions. For deterministic reductions, we prove that nothing interesting is reducible to MCSP via an oracle-independent reduction:

**Theorem 8.1.** *No language outside* P *can reduce to* MCSP *under polynomial-time Turing oracle-independent reductions. In other words, if a language $L$ polynomial-time-Turing-reduces to* $\mathrm{MCSP}^A$ *for any oracle $A$, then $L \in \mathsf{P}$; it can be also simply stated as*

$$\bigcap_A \mathsf{P}^{\mathrm{MCSP}^A} = \mathsf{P}.$$

In contrast to the results presented in Chapter 7 showing that NP-hardness of MCSP implies surprising consequences (*e.g.* $\mathsf{ZPP} \neq \mathsf{EXP}$), we emphasize that this theorem gives us an inherent limitation of a deterministic oracle-independent reduction. One implication is that NP-hardness of MCSP cannot be shown via a deterministic oracle-independent reduction unless $\mathsf{P} = \mathsf{NP}$.

We note that this precisely captures the limit of what we can deterministically reduce to MCSP. Indeed, currently no (nontrivial) deterministic reduction to MCSP is known at all unconditionally. The theorem suggests one reason behind this fact: in order to construct a deterministic reduction to MCSP, we need to use a property of MCSP that cannot be generalized to $\mathrm{MCSP}^A$ for some $A$, which appears very difficult due to our few knowledge about nonrelativizing circuit lower bounds.

It should be also noted that Theorem 8.1 implies that there exists an oracle $A$ such that $\mathrm{MCSP} \not\leq_{\mathrm{T}}^{\mathsf{P}} \mathrm{MCSP}^A$ unless $\mathrm{MCSP} \in \mathsf{P}$. At first glance (mainly due to its notation), it might be counterintuitive that an oracle version $\mathrm{MCSP}^A$ becomes "easier" than MCSP. The point is that the oracle $A$ in the notation $\mathrm{MCSP}^A$ refers to the fact that a circuit that is minimized has oracle access to $A$, but this does not necessarily increase the computational difficulty of minimizing such an $A$-oracle circuit.

Indeed, we exploit this fact to prove Theorem 8.1. Roughly speaking, for any oracle-independent reduction to MCSP, we adversarially choose an oracle $A$ so that any query that the reduction makes has circuit complexity of $O(\log n)$. Specifically, let $T_1, \ldots, T_{n^{O(1)}}$ be the truth tables queried by the reduction (on some computation path); we encode these truth tables into $A$ so that the truth table of $A(i, \text{-})$ is equal to $T_i$ for any $i$. For this oracle, the reduction cannot query any truth table that has high circuit complexity (relative to oracle $A$) because the size of the circuit that outputs $A(i, x)$ on input $x$ is $O(\log n)$ for any $i$. We then simulate the reduction by exhaustively[1] search small circuits of size up to $O(\log n)$.

We also prove that even randomized oracle-independent reduction is not sufficient to establish NP-hardness of MCSP:

**Theorem 8.2.** *If a language $L$ is reducible to* MCSP *via an oracle-independent randomized reduction with negligible error that makes at most one query, then*

[1] When the "size" of a circuit refers to the number of its wires, we cannot enumerate all such circuits in polynomial time since there are $O(\log n)^{O(\log n)} = n^{O(\log \log n)}$ possible circuits of size less than $O(\log n)$, which gives only a weak upper bound. We will thus regard the "size" of a circuit as its description length, and also require that we can encode a truth table into an oracle efficiently.

$L \in \mathsf{AM} \cap \mathsf{coAM}$. *In other words,*

$$\bigcap_A \mathsf{BPP}^{\mathrm{MCSP}^A[1]} \subseteq \mathsf{AM} \cap \mathsf{coAM}.$$

Here, $\mathsf{BPP}^{B[1]}$ denotes the class of languages reducible to an oracle $B$ via a randomized reduction with negligible error that makes at most one query.

In particular, $\bigcap_A \mathsf{BPP}^{\mathrm{MCSP}^A[1]}$ does not contain $\mathsf{NP}$ unless $\mathsf{NP} \subseteq \mathsf{coAM}$ (and in particular the polynomial hierarchy collapses [BHZ87]). Therefore, it is impossible to establish $\mathsf{NP}$-hardness of MCSP via such reductions (unless the polynomial hierarchy collapses).

### 8.1.1 Oracle-independent Reductions vs. Relativization

We note that an oracle-independent reduction is different from simple relativization. In a relativization setting, Ko [Ko91] showed the existence of a relativized world where MCSP is an $\mathsf{NP}$-intermediate problem: MCSP is neither in $\mathsf{coNP}$ nor is $\mathsf{NP}$-complete under polynomial-time Turing reductions. Specifically, he constructed an oracle $A$ such that $\mathsf{NP}^A$ is not contained in $\mathsf{P}^{\mathrm{MCSP}^A, A}$, thereby showing a relativized world where MCSP cannot be $\mathsf{NP}$-hard under polynomial-time Turing reductions. This shows the computational limit of MCSP in a relativized world.

In contrast, we discuss the computational limit of MCSP in a *real world* when MCSP is used by oracle-independent reductions. Technically, by exploiting the fact that $\mathsf{NP}$-machines have oracle access, Ko [Ko91] constructed an oracle $A$ so that some $\mathsf{NP}^A$-computation would go beyond the class $\mathsf{P}^{\mathrm{MCSP}^A, A}$. On the other hand, we construct an oracle $A$ so that $\mathsf{P}^{\mathrm{MCSP}^A}$-computation cannot be strong; in fact, it is essentially the same as $\mathsf{P}$.

### Organization

The rest of this chapter is organized as follows. In Section 8.2, we introduce some notation and the definition of circuit complexity. In Section 8.3, we observe that the known reductions to MCSP are oracle-independent. We prove Theorem 8.1 and 8.2 in Section 8.4 and 8.5, respectively.

## 8.2 Preliminaries

It is convenient to introduce the notion of *finite oracle* for diagonalization arguments.

**Definition 8.3.** *1. We say that $A_0$ is a* finite oracle *if $A_0 \colon \{0,1\}^* \to \{0,1,\bot\}$ and $A_0(x) = \bot$ for all but finitely many strings $x \in \{0,1\}^*$, where $\bot$ means "undefined."*

*2. For an oracle $A \subseteq \{0,1\}^*$ and a finite oracle $A_0$, we say that $A$ is* consistent *with $A_0$ if $A(x) = A_0(x)$ for any $x \in \{0,1\}^*$ such that $A_0(x) \neq \bot$.*

*3. Similarly, for $l \in \mathbb{N}$, we say that $A$ and $A_0$ are* consistent up to length $l$ *if it holds that $A(x) = 1$ if and only if $A_0(x) = 1$ for all strings $x \in \{0,1\}^*$ of length at most $l$.*

### 8.2.1 Definition of Circuit Size

For some technical reasons, we regard the description length of a circuit as its size. Thus we borrow notation of Kolmogorov complexity and write

$$\mathrm{K}_I(x) := \min\{\, |d| \mid I(d) = x \,\}$$

for every machine $I$ and every string $x \in \{0,1\}^*$. Throughout this chapter we will use a specific interpreter $I$ that is defined below.

We first fix our standard (oracle) circuit interpreter. We assume any standard way to encode circuits by binary strings. Note that a circuit may be an oracle circuit that can use oracle gates outputting $A(z)$ for a given input $z$ to the gate when a circuit is used with oracle $A$. Let $I_0$ denote a circuit interpreter for this encoding: that is, for any oracle $A$ and a given description $d$ of an oracle circuit $C$, the interpreter $I_0^A(d)$ yields the truth table of $C^A$. (Thus, $|I_0^A(d)| = 2^n$ for some $n$ and $I_0^A(d) = C^A(\underline{1}_n) \cdots C^A(\underline{2^n}_n)$.)

We will use the following facts that the standard circuit interpreter $I_0^A$ should have:

1. $I_0^A(d)$ is computable in time polynomial in $|d|$ and $|I_0^A(d)|$, given oracle access to $A$.

2. For all but finitely many truth tables $T \in \{0,1\}^*$ (where $|T|$ is a power of 2), there exists a circuit description of size less than $|T|^2$: that is, $\mathrm{K}_{I_0}(T) < |T|^2$.

3. Any oracle circuit $C$ whose description length is at most $m$ cannot query to an oracle any string of length greater than $m$. Thus, the output of $C^A$ only depends on the membership in $A$ of strings of length at most $m$.

We modify the standard circuit interpreter $I_0$ so that we can describe some type of circuits succinctly. For any $n \in \mathbb{N}$ and $d \in \{0,1\}^*$, let $C^A_{n,d}(x)$ be an oracle circuit that computes $A(x,d)$ (*i.e.* $A(\langle x,d\rangle)$) for a given input $x \in \{0,1\}^n$, by using a single oracle gate with input $\langle x,d\rangle$.

**Definition 8.4.** *Define an interpreter $I^A$ as follows:*

$$I^A(0d) := I_0^A(d),$$
$$I^A(1^n,d) := I_0^A(C^A_{n,d}) = A(\underline{1}_n,d)A(\underline{2}_n,d)\cdots A(\underline{2^n}_n,d),$$

*for any $n \geq 1$ and $d \in \{0,1\}^*$. For the other strings $d$ (e.g. $d = 1101$), leave $I^A(d)$ undefined.*

*For $A = \varnothing$, we write $I$ instead of $I^\varnothing$.*

**Remark.**

1. Recall that $\langle 1^n,d\rangle = 1^n01^nd$; hence $I^A$ is well-defined. Also, the definition of $I^A$ ensures that the description length of a circuit $C^A_{n,d}$ is at most $|\,\langle 1^n,d\rangle\,| = 2n+|d|+1$, which is exactly equal to the length of a query $\langle \underline{i}_n,d\rangle$ to oracle $A$.

2. For $A = \varnothing$, we have $\mathrm{K}_I(x) = \mathrm{K}_{I_0}(x) + 1$ for any $x \in \{0,1\}^* \setminus \{0\}^*$; hence, there is essentially no difference between our circuit complexity measure $\mathrm{K}_I(x)$ and a standard description length $\mathrm{K}_{I_0}(x)$.

3. For a general oracle $A$, since we assumed that the circuit $C^A_{n,d}$ can be described succinctly, we cannot guarantee that minimizing our complexity measure $\mathrm{K}_{I^A}$ is computationally equivalent to minimizing standard circuit complexity. However, all of the previous work (*e.g.* [Ko91, AHK17]) that we are aware of holds under our encoding scheme.

We define the minimum oracle circuit size problem $\mathrm{MCSP}^A$ by using $I^A$ as a circuit interpreter:

**Definition 8.5.** *The* minimum oracle circuit size problem $\mathrm{MCSP}^A$ *relative to an oracle* $A \subseteq \{0,1\}^*$ *takes a truth table* $T \in \{0,1\}^*$ *and a size-parameter* $s \in \mathbb{N}$*, and decides if* $\mathrm{K}_{I^A}(T) \leq s$*.*

## 8.3 Why Are the Known Reductions Oracle-independent?

In this section, we argue that the known reductions to MCSP are oracle-independent. We observe that the existing reductions only exploit (as a circuit lower bound) the fact that many truth tables require high (unrelativized) circuit complexity. Indeed, it is easy to observe that, for every oracle $A$, a set $B := \{\, x \in \{0,1\}^* \mid \mathrm{K}_{I^A}(x) \geq |x|^{1/2} \,\}$ defines a natural property useful against $\mathsf{SIZE}(2^{\Omega(n)})$. Indeed, by a simple counting argument, most of truth tables are not in $B$. Thus any reduction to a natural property can be seen as an oracle-independent reduction to MCSP (*e.g.* the reduction from inverting an auxiliary-input one-way function to a natural property Theorem 3.7).

Next, we show that an oracle-independent one-query reduction to MCSP allows us to convert a randomized algorithm with two-sided error into a randomized algorithm with zero-sided error. Moreover, the error probability is negligible.

**Theorem 8.6** (Kabanets and Cai [KC00])**.** $\mathsf{BPP} \subseteq \bigcap_A \mathsf{ZPP}^{\mathrm{MCSP}^A[1]}$.

*Proof Sketch.* Pick a truth table $T$ uniformly at random. By making a query to $\mathrm{MCSP}^A$, check if $\mathrm{K}_{I^A}(T) = n^{\Omega(1)}$. (Note that this also implies that $\mathrm{K}_I(T) = n^{\Omega(1)}$.) Now, if we successfully found a truth table $T$ that requires high circuit complexity, then we can use the pseudorandom generator of Impagliazzo and Wigderson [IW97] to derandomize a $\mathsf{BPP}$ computation. □

Finally, we observe that the reductions of [AGvM$^+$18] are oracle-independent. They presented a reduction from the rigid graph isomorphism problem to MKTP. We can capture KT-complexity by our notation by defining a circuit interpreter $I_0^A$ as follows: On input $1^t 0 d$, run the universal Turing machine $U^{A,d}(i)$ for each $i \geq 1$ one by one in time at most $t$. Let $n$ be the minimum $i$ such that $U^{A,d}(i)$ outputs $\perp$. Output the concatenation of $U^{A,d}(1), \cdots, U^{A,d}(n-1)$. This definition ensures that $\mathrm{K}_{I_0^A}(x) = \mathrm{KT}^A(x)+1$, and that $\mathrm{K}_{I^A}(x) \leq \mathrm{K}_{I_0^A}(x)+1 = \mathrm{KT}^A(x)+2$.

For this particular interpreter $I^A$, we prove:

**Theorem 8.7** ([AGvM$^+$18])**.** *For any oracle* $A$*, the rigid graph isomorphism problem is reducible to* $\mathrm{MCSP}^A$ *via a one-query* $\mathsf{BPP}$*-reduction.*

*Proof Sketch.* We only observe why their reduction still works for $\mathrm{MCSP}^A$, where $A$ denotes an arbitrary oracle $A$.

Given two graphs $(G_0, G_1)$, they constructed a string $x'$ whose length is a power of 2 and a threshold $\theta$ that satisfy the following: If the graphs are isomorphic, then $\mathrm{KT}(x') \ll \theta$ with probability 1. If the graphs are rigid and not isomorphic, then $x'$ contains information about a uniformly chosen random string of length at least $\theta$, and hence $\mathrm{KT}(x') \geq \mathrm{K}_U(x') \gg \theta$ with high probability. (Here, $\mathrm{K}_U(x')$ denotes the *time-unbounded* Kolmogorov complexity.)

Now consider an arbitrary oracle $A$. We claim that the rigid graph isomorphism problem reduces to checking if $(x', \theta) \in \mathrm{MCSP}^A$. Suppose that the graphs are isomorphic; in this case, we have $\mathrm{K}_{I^A}(x') \leq \mathrm{KT}^A(x') + 2 \leq \mathrm{KT}(x') + 2 \ll \theta$. On the other hand, suppose that the graphs are rigid and not isomorphic. Since

$x'$ contains information about a uniformly chosen random string, an information-theoretic argument shows that $\mathrm{K}_{U^A}(x') \gg \theta$ with high probability (even relative to $A$). By the universality of $U$, we have $\mathrm{K}_{U^A}(x') \leq \mathrm{K}_{I^A}(x') + O(1)$. Therefore, $\mathrm{K}_{I^A}(x') \geq \mathrm{K}_{U^A}(x') - O(1) \gg \theta$. □

To summarize, on one hand, relativization does not increase circuit complexity ($\mathrm{K}_{I^A}(x') \leq \mathrm{K}_I(x')$); on the other hand, we are prone to rely on the fact that a uniformly chosen random string requires high circuit complexity, which remains true for any $\mathrm{MCSP}^A$.

We mention that, for a specific oracle $A$, an efficient reduction to $\mathrm{MCSP}^A$ is known. Allender, Buhrman, Koucký, van Melkebeek and Ronneburger [ABK+06b] showed that $\mathsf{PSPACE} \subseteq \mathsf{ZPP}^{\mathrm{MCSP}^{\mathsf{QBF}}}$. Since their proof relies on the fact that $\mathsf{QBF}$ is $\mathsf{PSPACE}$-complete, the proof cannot be generalized to a reduction to MCSP; hence, their reduction cannot be regarded as an oracle-independent reduction to MCSP.

## 8.4 Limits of Oracle-independent Turing Reductions to MCSP

We show upper bounds for classes of languages that reduce to MCSP in an oracle-independent manner (*i.e.* in a way that one does not use a property of MCSP rather than that of a relativized version $\mathrm{MCSP}^A$). For example, we consider a situation where a language $L$ is reducible to $\mathrm{MCSP}^A$ for any $A$ via a polynomial-time Turing reduction; more precisely, for every $A$, there exists a polynomial-time Turing reduction from $L$ to $\mathrm{MCSP}^A$, *i.e.* $L \in \bigcap_A \mathsf{P}^{\mathrm{MCSP}^A}$. That is, only properties common to $\mathrm{MCSP}^A$ for any oracle $A$ are used to show that $L$ is in $\mathsf{P}^{\mathrm{MCSP}^A}$. We would like to show that $L$ is relatively easy in such situations.

In fact, we can indeed show that any language $L$ in $\bigcap_A \mathsf{P}^{\mathrm{MCSP}^A}$ is in $\mathsf{P}$.

**Theorem 8.8** (Restatement of Theorem 8.1)**.** *Let $L \subseteq \{0,1\}^*$ be a language such that for any oracle $A$, there exists a polynomial-time Turing reduction from $L$ to $\mathrm{MCSP}^A$. Then $L$ is in $\mathsf{P}$. In short, $\bigcap_A \mathsf{P}^{\mathrm{MCSP}^A} = \mathsf{P}$.*

We will prove this theorem as follows: We will argue that, for each polynomial-time reduction $M$, we can adversarially choose an oracle $A_M$ so that the reduction $M$ cannot query any truth table of high circuit complexity (by encoding the truth tables queried by $M$ into the oracle $A_M$). However, the assumption of the theorem states that a reduction $M$ can depend on an oracle $A$, and hence $A$ cannot depend on $M$. We first get around this difficulty by swapping the order of quantifiers: we reduce our theorem to the following lemma, in which a machine $M$ cannot depend on $A$.

**Lemma 8.9.** *Let $L \subseteq \{0,1\}^*$ be a language and $A_0$ be an arbitrary finite oracle. Suppose that there exists a polynomial-time oracle Turing machine $M$ such that $M^{\mathrm{MCSP}^A}(x) = L(x)$ for any $x \in \{0,1\}^*$ and any oracle $A$ consistent with $A_0$. Then, $L \in \mathsf{P}$.*

Note that, in this lemma, a *single* machine $M$ is required to compute $L$ with respect to *every* oracle version $\mathrm{MCSP}^A$. We will later prove this lemma by choosing, for each reduction $M$ and input $x$, an oracle $A_{M,x}$ so that the reduction $M$ to $\mathrm{MCSP}^{A_{M,x}}$ can be simulated in polynomial time. Before its proof, we show that Lemma 8.9 implies Theorem 8.1 by using a simple diagonalization argument.

*Proof of Theorem 8.1 based on Lemma 8.9.* We prove the contrapositive: Assuming $L \notin \mathsf{P}$, the aim is to construct an oracle $A$ such that $L \notin \mathsf{P}^A$. Such

an oracle $A = \bigcup_e B_e$ is constructed in stages. Let all the polynomial-time oracle Turing machines be $\{M_1, M_2, \cdots\}$.

At stage $e$, we construct a finite oracle $B_e$. At stage 0, set $B_0(y) := \perp$ for all $y \in \{0,1\}^*$. At stage $e \geq 1$, we apply Lemma 8.9 for $M = M_e$ and $A_0 = B_{e-1}$: by the assumption that $L \notin \mathsf{P}$, there exist some string $x_e$ and some oracle $B_e$ consistent with $B_{e-1}$ such that $M_e^{\mathrm{MCSP}^{B_e}}(x_e) \neq L(x_e)$. We may assume that $B_e$ is a finite oracle: indeed, since the computation of $M_e^{\mathrm{MCSP}^{B_e}}$ on input $x_e$ makes a finite number of queries to $\mathrm{MCSP}^{B_e}$, the answers of the queries also depend on a finite portion of $B_e$. Define an oracle $A$ as the union of all the oracles $B_e$ whose $\perp$ is replaced by 0.

Since $A$ is consistent with $B_e$, it holds that $M_e^{\mathrm{MCSP}^{B_e}}(x_e) = M_e^{\mathrm{MCSP}^{A}}(x_e)$ for each $e \geq 1$. By the definition of $x_e$, we have $M_e^{\mathrm{MCSP}^{B_e}}(x_e) \neq L(x_e)$. Therefore, $M_e^{\mathrm{MCSP}^{A}}(x_e) \neq L(x_e)$ holds for any $e$, and hence $L \notin \mathsf{P}^{\mathrm{MCSP}^{A}}$. □

Now we give a proof of Lemma 8.9. The idea is as follows: For any reduction $M$ and any input $x$, we simulate the reduction $M$ by answering $M$'s query by exhaustively searching all the circuits of size at most $O(\log n)$. On this specific computation path of $M$, we claim that there exists some oracle $A_{M,x}$ such that the simulated computation path coincides with the computation path of the reduction $M$ to $\mathrm{MCSP}^{A_{M,x}}$, thereby showing that the output of the simulation of $M$ is $L(x)$: Since $M$ is a polynomial-time machine, the number of the queries on the computation path is at most $n^{O(1)}$. Thus, the index $i$ of the queries can be described in $O(\log n)$ bits, and hence the description length of the oracle circuit $C^{A_{M,x}}(j) := A_{M,x}(j,i)$ is at most $O(\log n)$. By defining $A_{M,x}(j,i) := T_{ij}$ for each truth table $T_i$ queried by $M$, any truth table $T_i$ admits a circuit of size at most $O(\log n)$.

Let us turn to a formal proof. Let $M$ be a polynomial-time oracle machine that computes $L$ given oracle access to $\mathrm{MCSP}^A$ in time $n^c$ for some constant $c$, where $A$ denotes an arbitrary oracle consistent with $A_0$. We define a polynomial-time machine $M_0$ that simulates $M$ without using $\mathrm{MCSP}^A$ as follows: On input $x \in \{0,1\}^*$ of length $n$, simulate $M$ on input $x$, and accept if and only if $M$ accepts. If $M$ makes a query $(T, s)$, then we try to compute the circuit complexity $\mathrm{K}_{I^{A_0}}(T)$ of the truth table $T$ relative to a finite oracle $A_0$, by an exhaustive search up to size at most $4c \log n$. (More specifically, we compute the shortest description $d$ of length at most $4c \log n$ such that $I^{A_0}(d) = T$, where we regard $A_0 \subseteq \{0,1\}^*$ as an oracle by replacing $\perp$ by 0 in finite oracle $A_0$.) If the circuit complexity $\mathrm{K}_{I^{A_0}}(T)$ has turned out to be greater than $4c \log n$, then define $s' := 4c \log n$; otherwise define $s' := \mathrm{K}_{I^{A_0}}(T)$ $(\leq 4c \log n)$. (*i.e.* $s' := \min\{4c \log n, \mathrm{K}_{I^{A_0}}(T)\}$.) Answer "Yes" to the query if and only if $s' \leq s$.

It is easy to see that $M_0$ is indeed a polynomial-time machine, since there are only $2^{O(\log n)}$ circuits of size at most $O(\log n)$. (Recall that we regard a circuit size as a description length.) Thus, it is sufficient to prove the following:

**Claim 8.10.** *For all sufficiently large $n$ and all inputs $x$ of length $n$, there exists an oracle $A_{M,x}$ consistent with $A_0$ such that $M_0(x) = M^{\mathrm{MCSP}^{A_{M,x}}}(x)$.*

Note that the assumption of Lemma 8.9 implies that $M^{\mathrm{MCSP}^{A_{M,x}}}(x) = L(x)$. Thus, the claim implies that $M_0(x) = L(x)$ and hence $L \in \mathsf{P}$.

*Proof of Claim 8.10.* Fix $n$ sufficiently large and an input $x \in \{0,1\}^n$. For $i \in [n^c]$, let $T_i$ be the truth table in the $i$th query that $M$ makes on the computation path simulated by $M_0$ on input $x$.

We define an oracle $A_{M,x} = A$ as follows (here, $A_{M,x}$ is abbreviated as $A$ for notational convenience): For any string $q \in \{0,1\}^*$ of length less than $4c \log n$, define $A(q) = 1$ if and only if $A_0(q) = 1$. For strings of length $4c \log n$, we encode $T_i$ into oracle $A$ so that the circuit complexity of $T_i$ relative to $A$ is at most $4c \log n$: Specifically, we would like to define a description $d_i$ of length (*exactly* equal to) $4c \log n$ so that $I^A(d_i) = T_i$. To this end, let $a_i := \log |T_i|$ and define $d_i := \langle 1^{a_i}, \underline{i}_{k_i} \rangle$, where $k_i \in \mathbb{N}$ is defined so that $|d_i| = 2a_i + 1 + k_i = 4c \log n$. Here, $\underline{i}_{k_i}$ is well-defined: indeed, we have $a_i = \log |T_i| \leq c \log n$, which implies that $k_i := 4c \log n - 2a_i - 1 \geq c \log n$, and thus $i \leq 2^{c \log n} \leq 2^{k_i}$. Now define $A(\underline{j}_{a_i}, \underline{i}_{k_i}) := T_{ij}$ for each $j \in [2^{a_i}]$. By the definition of $I^A$, the truth table $T_i$ can be described succinctly: $I^A(d_i) = A(\underline{1}_{a_i}, \underline{i}_{k_i}) \cdots A(\underline{2^{a_i}}_{a_i}, \underline{i}_{k_i}) = T_i$; thus, the circuit complexity $\mathrm{K}_{I^A}(T_i)$ of $T_i$ is at most $|d_i| = 4c \log n$.

It remains to show that, for each query $(T_i, s)$ that $M$ makes on the computation path simulated by $M_0$, circuit complexity $s'$ $(= \min\{4c \log n, \mathrm{K}_{I^{A_0}}(T_i)\})$ calculated by $M_0$ coincides with $\mathrm{K}_{I^A}(T_i)$; note that this implies that $M_0(x) = M^{\mathrm{MCSP}^A}(x)$, because the computation path simulated by $M_0$ coincides with that of $M$ relative to $\mathrm{MCSP}^A$. In order to see $\mathrm{K}_{I^A}(T_i) = \min\{4c \log n, \mathrm{K}_{I^{A_0}}(T_i)\}$, first we note that $A$ and $A_0$ are consistent up to length $4c \log n - 1$; thus, for small circuits, circuit complexity relative to $A$ remains the same with circuit complexity relative to $A_0$, because small circuits cannot query long strings of length $4c \log n$. Formally, suppose that $\mathrm{K}_{I^{A_0}}(T_i) < 4c \log n$ (*i.e.* $s' = \mathrm{K}_{I^{A_0}}(T_i)$). In this case, there exists some description $d$ of length less than $4c \log n$ such that $I^{A_0}(d) = T_i$. Since the circuit described by $d$ cannot make any query of length greater than $|d|$, it holds that $I^{A_0}(d) = I^A(d)$. Thus $\mathrm{K}_{I^A}(T_i) \leq \mathrm{K}_{I^{A_0}}(T_i) < 4c \log n$. Similarly, we have $\mathrm{K}_{I^{A_0}}(T_i) \leq \mathrm{K}_{I^A}(T_i)$, and hence $\mathrm{K}_{I^A}(T_i) = \mathrm{K}_{I^{A_0}}(T_i) = s'$. Now suppose that $\mathrm{K}_{I^{A_0}}(T_i) \geq 4c \log n$ (*i.e.* $s' = 4c \log n$). We claim that $\mathrm{K}_{I^A}(T_i) = 4c \log n$. Since we have $\mathrm{K}_{I^A}(T_i) \leq 4c \log n$ by the definition of $A$, it is sufficient to show that $\mathrm{K}_{I^A}(T_i) < 4c \log n$ is not true. Assume, by way of contradiction, that $\mathrm{K}_{I^A}(T_i) < 4c \log n$. By the same argument above, it must be the case that $\mathrm{K}_{I^A}(T_i) \geq \mathrm{K}_{I^{A_0}}(T_i) \geq 4c \log n$, which is a contradiction. □

This completes the proof of Lemma 8.9.

**Remark.** If we regard a size of a circuit as the number of its wires, then the upper bound $\mathsf{P}$ becomes $\mathsf{DTIME}(n^{O(\log \log n)})$. Specifically, let $\mathrm{MCSP}'^A$ denotes a version of $\mathrm{MCSP}^A$ in which a size of a circuit is measured by the number of its wires. Then we have $\bigcap_A \mathsf{P}^{\mathrm{MCSP}'^A} \subseteq \mathsf{DTIME}(n^{O(\log \log n)})$. This can be proved by simply changing $M_0$ in the proof above so that $M_0$ exhaustively search all the circuits of at most $O(\log n)$ *wires* in time $O(\log n)^{O(\log n)} = n^{O(\log \log n)}$.

## 8.5 Limits of Oracle-independent Randomized Reductions to MCSP

In this section, we discuss the limits of a randomized reduction to MCSP that can be generalized to a reduction to $\mathrm{MCSP}^A$ for an arbitrary oracle $A$. Our focus is a randomized reduction with negligible two-sided error that can make at most one query:

**Definition 8.11.** *Let $L, B \subseteq \{0,1\}^*$ be a language and an oracle, respectively. We say that $L$ reduces to $B$ via a* one-query $\mathsf{BPP}$-reduction *and write $L \in \mathsf{BPP}^{B[1]}$ if there exist polynomial-time machines $M, Q$ and a negligible function $\epsilon$ such that,*

*for any* $x \in \{0,1\}^*$,

$$\Pr_{r \in \{0,1\}^{|x|^{O(1)}}}[M(x, r, B(Q(x, r))) = L(x)] \geq 1 - \epsilon(|x|).$$

*Here, we say that a function* $\epsilon$ *is negligible if for all polynomials* $p$*, for all sufficiently large* $n \in \mathbb{N}$*, the function is bounded by the inverse of* $p$*: that is,* $\epsilon(n) < \frac{1}{p(n)}$.

Note that we require the error probability to be negligible. Since the number of queries is restricted to one, we cannot apply the standard error-reduction argument; hence, this definition may be stronger than a definition whose error probability is a constant. We leave as an open problem improving our result to the case when the error probability is a constant.

We prove that there is no language outside $\mathsf{AM} \cap \mathsf{coAM}$ that can reduce to $\mathrm{MCSP}^A$ for an arbitrary oracle $A$ via a one-query randomized reduction:

**Theorem 8.12** (Restatement of Theorem 8.2)**.** *Let* $L \subseteq \{0,1\}^*$ *be a language such that for any oracle* $A$*, there exists a one-query* $\mathsf{BPP}$*-reduction from* $L$ *to* $\mathrm{MCSP}^A$*. Then* $L$ *is in* $\mathsf{AM} \cap \mathsf{coAM}$*. In short,*

$$\bigcap_A \mathsf{BPP}^{\mathrm{MCSP}^A[1]} \subseteq \mathsf{AM} \cap \mathsf{coAM}.$$

As with Theorem 8.1, we first swap the order of quantifiers. However, in order to swap the order of quantifiers, we need to enumerate all the negligible functions, which is not countably many; thus, we sidestep this by requiring that the error probability is an inverse polynomial $1/q$ in the running time of machines $M$ and $Q$. Also, since a one-query $\mathsf{BPP}$-reduction is closed under complement, we only have to show that the target language is in $\mathsf{AM}$.

**Lemma 8.13.** *There exists some universal polynomial* $q$ *(specified later) that satisfies the following: Let* $L, A_0$ *be a language and a finite oracle, respectively. Suppose that there exist a polynomial* $p$ *and Turing machines* $M, Q$ *such that* $M$ *and* $Q$ *run in time* $p(n)$ *and*

$$\Pr_{r \in \{0,1\}^{p(n)}}[M(x, r, \mathrm{MCSP}^A(Q(x, r))) = L(x)] \geq 1 - \frac{1}{q(p(n))}$$

*for any* $x \in \{0,1\}^*$ *of length* $n$ *and any oracle* $A \subseteq \{0,1\}^*$ *consistent with* $A_0$*. Then, we have* $L \in \mathsf{AM}$*.*

We prove that Lemma 8.13 implies Theorem 8.2:

*Proof of Theorem 8.2 based on Lemma 8.13.* We prove the contrapositive: Assuming $L \notin \mathsf{AM}$, we will construct an oracle $A$ such that $L \notin \mathsf{BPP}^{\mathrm{MCSP}^A[1]}$ by diagonalization.

Enumerate all the tuples $\{(M_e, Q_e, c_e)\}_{e \geq 1}$, where $M_e$ and $Q_e$ are polynomial-time machines and $c_e \in \mathbb{N}$. We assume that, for each tuple $(M_e, Q_e, c_e)$, there exist infinitely many $e' \in \mathbb{N}$ such that $(M_e, Q_e, c_e) = (M_{e'}, Q_{e'}, c_{e'})$.

At stage $e \geq 1$, we construct a finite oracle $B_e$ that fools a one-query $\mathsf{BPP}$ reduction $(M_e, Q_e)$ that runs in time $n^{c_e}$: If $M_e$ or $Q_e$ does not run in time $n^{c_e}$, then we define $B_e := B_{e-1}$. Otherwise, we can apply the contrapositive of Lemma 8.13 to $M_e$ and $Q_e$: there exist some input $x_e$ and some oracle $B_e$ consistent with $B_{e-1}$ such that $\Pr_r[M_e(x_e, r, \mathrm{MCSP}^{B_e}(Q_e(x_e, r))) = L(x_e)] < 1 - \frac{1}{q(n^{c_e})}$. We can make $B_e$ a finite oracle, since $M_e$ depends on only a finite

portion of $B_e$. This completes stage $e$. Define $A$ as the union of all the oracles $B_e$ whose $\perp$ is replaced by 0.

We claim that $L \notin \mathsf{BPP}^{\mathrm{MCSP}^A[1]}$. Assume otherwise. Then there exist a constant $c > 1$, a negligible function $\epsilon$, and machines $M$ and $Q$ that run in time $n^c$ such that

$$\Pr_r[M(x, r, \mathrm{MCSP}^A(Q(x, r))) = L(x)] \geq 1 - \epsilon(|x|) \tag{8.1}$$

for all $x \in \{0,1\}^*$. Fix a sufficiently large $n_0 \in \mathbb{N}$ such that $\epsilon(n) < \frac{1}{q(n^{c+1})}$ for all $n \geq n_0$. Let $M'$ be the Turing machine[2] that, on input $x$, outputs a hardwired answer $L(x)$ if $|x| \leq n_0$, and simulates $M$ otherwise. Note that the running time of $M'$ is at most $n^{c+1}$.

By the construction above, there exists $e \geq n_0$ such that $(M_e, Q_e, c_e) = (M', Q, c+1)$. By the definition of $x_e$, we have $\Pr_r[M'(x_e, r, \mathrm{MCSP}^A(Q(x_e, r))) = L(x_e)] < 1 - \frac{1}{q(|x_e|^{c+1})}$. Moreover, since $M'$ outputs a correct answer with probability 1 on input $x$ of length at most $n_0$, it holds that $|x_e| > n_0$; thus, we have $\epsilon(|x_e|) < \frac{1}{q(|x_e|^{c+1})}$; in addition, the machine $M'$ behaves in the same way with $M$. Hence, the success probability of $(M, Q)$ on input $x_e$ is equal to that of $(M', Q)$ on input $x_e$, which is bounded above by $1 - \frac{1}{q(|x_e|^{c+1})} < 1 - \epsilon(|x_e|)$. This contradicts (8.1). □

Now we outline the proof of Lemma 8.13.

We will first show that we may assume that all the queries that $Q$ makes have a truth table of a fixed length $2^t$ and a fixed size-parameter $s$ for some $t, s \in \mathbb{N}$. There is no loss of generality in assuming this because there are only polynomially many possibilities: the number of all the possible lengths of a truth table and size-parameters is at most $n^c$ for some $c$. Moreover, we may fix how to use the answer of a query: specifically, for a random choice $r$, define $f\colon \{0,1\} \to \{0,1\}$ (which has 4 possible choices) so that $f(b) = M(x, r, b)$. (For example, $f(b) = b$ means that $M$ accepts if and only if the query is a positive instance of $\mathrm{MCSP}^A$.)

We classify the set of random choices $r$ into $R_{f,t,s}$ according to these parameters $(f, t, s)$. If $x \in L$, then there must exist some $(f, t, s)$ such that $f(\mathrm{MCSP}^A(Q(x, r))) = 1$ with high probability over the choice of $r \sim R_{f,t,s}$. On the other hand, if $x \notin L$, then any $(f, t, s)$ must satisfy $f(\mathrm{MCSP}^A(Q(x, r))) = 0$ with high probability. Therefore, it is sufficient to prove that, for a specific $(f, t, s)$, there exists an $\mathsf{AM}$ protocol that checks if $f(\mathrm{MCSP}^A(Q(x, r))) = 1$ with high probability conditioning on $r \in R_{f,t,s}$.

Let us assume that $f(b) = b$ for simplicity. Then, it is sufficient to estimate the probability

$$P_{f,t,s} := \Pr_{r \sim R_{f,t,s}}[f(\mathrm{MCSP}^A(Q(x, r))) = 1] = \Pr_{r \sim R_{f,t,s}}[Q(x, r) \in \mathrm{MCSP}^A]$$

by an $\mathsf{AM}$ protocol. If the probability $P_{f,t,s}$ is close to 1, then the distribution induced by $Q(x, r)$ concentrates on a limited number of instances: indeed, since there are at most $2^{s+1}$ positive instances in $\mathrm{MCSP}^A$ for a size-parameter $s$, the query $Q(x, r)$ must be one of such instances with probability at least $P_{f,t,s}$. Conversely, suppose that the query distribution $Q(x, r)$ concentrates on a limited

[2] $M'$ can be implemented by a Turing machine as follows: Read the first $n_0 + 1$ bits of the input (if any). If the input length is at most $n_0$, then output the hardwired answer. Otherwise, move the head of the input tape to the initial position, and continue the computation of $M$. This implementation costs at most $2n_0$ additional steps.

number of instances $\{(T_1, s), (T_2, s), \cdots\}$; we may encode $T_i$ into an oracle $A$ and force these instances to be positive (*i.e.* $(T_i, s) \in \mathrm{MCSP}^A$); as a result, the probability $P_{f,t,s}$ is not small (since the instances $(T_i, s)$ are positive). Therefore, the task reduces to checking whether the query distribution concentrates on a limited number of instances.

To this end, we will use the heavy samples protocol [BT06b]. We say that an instance $(T, s)$ is $\beta$-heavy if the probability that $(T, s)$ is queried (*i.e.* $(T, s) = Q(x, r)$) is at least $\beta$. The heavy-sample protocol of Bogdanov and Trevisan [BT06b] allows us to estimate the probability that $Q(x, r)$ is $\beta$-heavy.

**Lemma 8.14** (follows from Theorem 5.21)**.** *Let $\mathcal{D} = \{\mathcal{D}_n\}_{n\in\mathbb{N}}$ be a polynomial-time samplable distribution. There exist a universal constant $c$ ($c = 2^{11}$ will do) and an $\mathsf{AM} \cap \mathsf{coAM}$ protocol that solves the following promise problem: Given input $1^n$ and a threshold $\beta \in [0, 1]$, accept if $\Pr_{y\sim\mathcal{D}_n}[\ y$ is $c\beta$-heavy $] \geq \frac{3}{4}$, and reject if $\Pr_{y\sim\mathcal{D}_n}[\ y$ is $\beta$-heavy $] \leq \frac{1}{4}$.*

Now we give a formal proof of Lemma 8.13. For the proof, we need to show an $\mathsf{AM}$ protocol deciding whether $x \in L$; we will show the protocol by a sequence of claims.

We begin with clarifying our setting and introducing some notation. Let $p(n)$ be a polynomial that is an upper bound of the running time of $M$ and $Q$. Fix a sufficiently large $n \in \mathbb{N}$ and an input $x \in \{0, 1\}^n$.

Let $f\colon \{0, 1\} \to \{0, 1\}$ be a function, and $t, s \in \mathbb{N}$. We define $R_{f,t,s} \subseteq \{0, 1\}^{p(n)}$ as the set of all the random choices $r \in \{0, 1\}^{p(n)}$ such that $M(x, r, b) = f(b)$ for all $b \in \{0, 1\}$ and $(T, s) = Q(x, r)$ and $|T| = 2^t$. That is, $f$ specifies how to use the answer from oracle $\mathrm{MCSP}^A$, and $t$ and $s$ specify the length of the truth table and the size-parameter in the query, respectively. Let $X := \{\, (f, t, s) \mid R_{f,t,s} \neq \varnothing \,\}$. We may assume, without loss of generality, that $s \leq p(n)^2$ as otherwise $Q(x, r)$ is obviously a positive instance; hence, $|X| \leq 2^2 \cdot \log p(n) \cdot p(n)^2 \leq 4p(n)^3$.

Define $P_{f,t,s} := \Pr_r[f(\mathrm{MCSP}^A(Q(x, r))) = 1 \mid r \in R_{f,t,s}]$ for $(f, t, s) \in X$. Let us divide the probability that $M$ accepts $x$ by conditioning on $r \in R_{f,t,s}$:

$$\Pr_r[M(x, r, \mathrm{MCSP}^A(Q(x, r)) = 1] = \sum_{(f,t,s)\in X} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s}. \tag{8.2}$$

Since there are polynomially many choices for $(f, t, s)$, there must be some $(f, t, s) \in X$ that can be used as a "witness" for $x \in L$ in our $\mathsf{AM}$ protocol. Specifically, the following claim holds:

**Claim 8.15.** *Let $\delta(n) := \sqrt{1/q(p(n))}$ and $\delta'(n) := 9p(n)^3\delta(n)$.*

1. *If $x \in L$, then there exists $(f, t, s) \in X$ such that $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$ and $P_{f,t,s} \geq 1 - \delta'(n)$.*
2. *If $x \notin L$, then $P_{f,t,s} \leq \delta(n)$ for all $(f, t, s) \in X$ such that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$.*

*Proof of Claim 8.15.*

1. Suppose that $x \in L$; then, the probability (8.2) is at least $1-\delta(n)^2$. Assume, by way of contradiction, that $P_{f,t,s} < 1-\delta'(n)$ for all $(f, t, s) \in X$ such that

$\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$. Then,

$$
\begin{aligned}
& 1-\delta(n)^2 \\
\leq & \sum_{(f,t,s)\in X} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \\
= & \sum_{\substack{(f,t,s)\in X \\ \Pr[r\in R_{f,t,s}]\geq 2\delta(n)}} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \quad + \sum_{\substack{(f,t,s)\in X \\ \Pr[r\in R_{f,t,s}]< 2\delta(n)}} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \\
\leq & 1-\delta'(n)+2|X|\delta(n) \leq 1-9p(n)^3\delta(n)+8p(n)^3\delta(n) = 1-p(n)^3\delta(n).
\end{aligned}
$$

Thus $p(n)^3 \leq \delta(n) < 1$, which is a contradiction.

2. Let $X'$ be the set of all $(f,t,s) \in X$ such that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$. Suppose that $x \notin L$; then,

$$\delta(n)^2 \geq \sum_{(f,t,s)\in X} \Pr_r[r \in R_{f,t,s}] \cdot P_{f,t,s} \geq \delta(n) \cdot \sum_{(f,t,s)\in X'} P_{f,t,s},$$

which clearly implies that $P_{f,t,s}$ is at most $\delta(n)$ for each $(f,t,s) \in X'$.

□

In our AM protocol, the prover first sends $(f,t,s)$ to the verifier; an honest prover is supposed to send $(f,t,s) \in X$ that satisfies the first condition in Claim 8.15 above. Then, what we need is to show a verifier of an AM protocol as stated in the following claim.

**Claim 8.16.** *There exists a verifier $V$ of an* AM *protocol such that, for a given $x \in \{0,1\}^*$ and $(f,t,s) \in X$,*

1. *if $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$ and $P_{f,t,s} \geq 1-\delta'(n)$, then $V$ accepts with high probability by communicating with some prover, and*
2. *if $\Pr[r \in R_{f,t,s}] < \delta(n)$ or $P_{f,t,s} \leq \delta(n)$, then $V$ rejects with high probability with any prover.*

We explain below how to define this verifier $V$. Recall that $1/\delta(n) = n^{O(1)}$; thus, it is easy to distinguish the case when $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$ and the case when $\Pr[r \in R_{f,t,s}] < \delta(n)$. The following claim states this formally.

**Claim 8.17.** *There exists a randomized polynomial-time algorithm that, given $x \in \{0,1\}^*$ and $(f,t,s) \in X$,*

1. *accepts with high probability if $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$, and*
2. *rejects with high probability if $\Pr[r \in R_{f,t,s}] < \delta(n)$.*

*Proof Sketch.* Sample $r_1, \cdots, r_m \sim \{0,1\}^{p(n)}$ uniformly at random for $m = O(n/\delta(n)^2)$. Accept if and only if the number of $i$'s such that $r_i \in R_{f,t,s}$ is at least $1.5 \cdot \delta(n)m$. By applying the Chernoff bound, this algorithm distinguishes the two cases with probability at least $1-2^{-n}$. □

Therefore in our AM protocol, verifier $V$ first uses this algorithm to check whether $\Pr[r \in R_{f,t,s}] \geq 2\delta(n)$ or $\Pr[r \in R_{f,t,s}] < \delta(n)$. If $\Pr[r \in R_{f,t,s}] < \delta(n)$

is confirmed by the algorithm, then $V$ can immediately reject $(f,t,s)$ and halt. Thus, it remains to design a part where $V$ determines whether $P_{f,t,s} \geq 1 - \delta'(n)$ or $P_{f,t,s} \leq \delta(n)$ ($\leq \delta'(n)$) holds, assuming that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$. Note that this assumption implies that the uniform distribution on $R_{f,t,s}$ can be sampled efficiently: indeed, sample $r \sim \{0,1\}^{p(n)}$ until we obtain an element $r$ such that $r \in R_{f,t,s}$; this sampling algorithm succeeds within $O(1/\delta(n))$ steps in expectation.

Now our task is to define an $\mathsf{AM}$ protocol determining whether $P_{f,t,s}$ is close to 1 or smaller than $\delta'(n)$, assuming that the query distribution $Q(x,r)$ where $r \sim R_{f,t,s}$ can be sampled efficiently. Note that $P_{f,t,s}$ may depend on $x$ and $\mathrm{MCSP}^A$. We will show that the task above can be reduced to checking whether a certain concentration occurs, by defining $A$ so that heavy queries become positive instances. Then we will check if such a concentration occurs by the heavy samples protocol of Lemma 8.14.

In order to reduce Claim 8.16 to the heavy samples protocol, we introduce some notation: Fix $(f,t,s) \in X$. Let us sort all the truth tables $\{T_1, \cdots, T_{2^{2^t}}\} = \{0,1\}^{2^t}$ of length $2^t$ in the order of heaviness: namely, let $p_i := \Pr_{r \sim R_{f,t,s}}[Q(x,r) = (T_i, s)]$ and $p_1 \geq p_2 \geq \cdots \geq p_{2^{2^t}}$. Let $p(I)$ denote $\sum_{i \in I} p_i$ for $I \subseteq [2^{2^t}]$. Define the set of $\alpha$-heavy indices (with respect to the distribution induced by $Q(x,r)$ where $r \sim R_{f,t,s}$) as $I_\alpha := \{\, i \in [2^{2^t}] \mid p_i \geq \alpha \,\}$ for $\alpha \geq 0$. Note that $p(I_\alpha) = \Pr_{r \sim R_{f,t,s}}[\, Q(x,r) \text{ is } \alpha\text{-heavy} \,]$. We also define $P_{\mathrm{id},t,s} := \Pr_{r \sim R_{f,t,s}}[\, Q(x,r) \in \mathrm{MCSP}^A \,]$.

We will show that the condition that $P_{\mathrm{id},t,s}$ is close to 1 is (almost) characterized by the fact that the query distribution is concentrated on $\{\, (T_i, s) \mid i \in I_\beta \,\}$, namely the set of $\beta$-heavy instances for some threshold $\beta > 0$.

**Claim 8.18.** *There exists an oracle $A$ consistent with $A_0$ up to length $7 \log p(n)$ that satisfies the following: for any $(f,t,s) \in X$ such that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$ and $s > 7 \log p(n)$ hold,*

1. *if $P_{\mathrm{id},t,s} \geq 1 - \delta'(n)$, then $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$, and*
2. *if $P_{\mathrm{id},t,s} \leq \delta'(n)$, then $p(I_\beta) \leq \delta''(n)$.*

*Here, we define $\delta''(n) := 2\,p(n)^7\,\delta'(n) = 18\,p(n)^{10}\,\delta(n)$ and $\beta := \delta''(n)\,2^{-s}$, and $c$ denotes the universal constant in Lemma 8.14.*

This claim allows us to apply the heavy samples protocol. Let us complete the proof of Claim 8.16 before proving Claim 8.18.

*Proof of Claim 8.16.* As explained above, it is sufficient to show that our verifier $V$ can check whether $P_{f,t,s} \geq 1 - \delta'(n)$ or $P_{f,t,s} \leq \delta'(n)$, assuming that $\Pr[r \in R_{f,t,s}] \geq \delta(n)$.

If $f \equiv 1$ or $f \equiv 0$, then the task is trivial: in the former case, $P_{f,t,s} = 1$ and hence $V$ may immediately accept; in the latter case, $V$ rejects.

If $s \leq 7 \log p(n)$, then we may decide whether $Q(x,r) \in \mathrm{MCSP}^{A_0}$ or not by an exhaustive search in time $2^{O(s)} = n^{O(1)}$. Since $A$ and $A_0$ are consistent up to length $7 \log p(n)$, as in the proof of Lemma 8.9, it holds that $Q(x,r) \in \mathrm{MCSP}^{A_0}$ if and only if $Q(x,r) \in \mathrm{MCSP}^A$. Therefore, we may estimate $P_{f,t,s}$ by sampling $r \sim R_{f,t,s}$ and then decide whether $Q(x,r) \in \mathrm{MCSP}^A$ by the exhaustive search.

Otherwise, we have $s > 7 \log p(n)$ and hence Claim 8.18 can be applied. Now suppose that $f(b) = b$ for any $b \in \{0,1\}$. In this case, it holds that $P_{f,t,s} = P_{\mathrm{id},t,s}$; thus Claim 8.18 states that, if $P_{f,t,s} \geq 1 - \delta'(n)$ then $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$, and if $P_{f,t,s} \leq \delta'(n)$ then $p(I_\beta) \leq \delta''(n)$. Now we may apply the heavy samples protocol

for the query distribution induced by $Q(x, r)$ where $r \sim R_{f,t,s}$, in order to check whether $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$ or $p(I_\beta) \leq \delta''(n)$: more specifically, $V$ accepts in the former case by running the AM protocol of Lemma 8.14.

Similarly, if $f(b) = 1-b$, then we have $P_{f,t,s} = 1-P_{\mathrm{id},t,s}$. This implies the same condition except for flipping YES and NO: if $P_{f,t,s} \geq 1-\delta'(n)$ then $p(I_\beta) \leq \delta''(n)$; if $P_{f,t,s} \leq \delta'(n)$ then $p(I_{c\beta}) \geq 1-3c\delta''(n)$. Thus, we may apply the heavy samples protocol to check whether $p(I_\beta) \leq \delta''(n)$ or $p(I_{c\beta}) \geq 1 - 3c\delta''(n)$: specifically, $V$ accepts in the former case by running the coAM protocol of Lemma 8.14.

Note that we may pick the polynomial $q$ that specifies the error probability so that $3c\delta''(n) \leq \frac{1}{4}$ (which allows us to use Lemma 8.14): indeed, if we define $q(n) := O(n^{22})$ then we have $\delta(n) = \sqrt{1/q(p(n))} = O(p(n)^{-11})$ and hence $3c\delta''(n) = O(p(n)^{10}\,\delta(n)) = o(1)$. □

All that remains is to show Claim 8.18. The intuition is as follows: Suppose that the probability that a positive instance is queried is large (*i.e.* $P_{\mathrm{id},t,s} \geq 1 - \delta'(n)$). Since there are at most $2^{s+1}$ truth tables that have circuit complexity at most $s$, the query distribution must concentrate on such positive instances; thus $p([2^{s+1}])$ is also large, which in particular implies that $p(I_{c\beta})$ is large (since $\beta$ is in fact chosen so that $p(I_\beta)$ is roughly equal to $p([2^{s+1}])$).

Conversely, suppose that the query distribution concentrates on heavy instances $\{(T_1, s), \ldots, (T_{2^k}, s)\}$ (*i.e.* $p([2^k])$ is large) for some $k$. In this case, we may encode the heavy truth tables into the oracle $A$; thereby we can force these truth tables to be positive instances, which implies that $p([2^k]) \leq P_{\mathrm{id},t,s}$; hence $P_{\mathrm{id},t,s}$ cannot be small. The details follow:

*Proof of Claim 8.18.* 1. Suppose that $P_{\mathrm{id},t,s} \geq 1 - \delta'(n)$. Then,

$$1 - \delta'(n) \leq \Pr_{r \sim R_{f,t,s}}[Q(x, r) \in \mathrm{MCSP}^A]$$
$$= \sum_{i\colon (T_i,s) \in \mathrm{MCSP}^A} p_i \leq \sum_{i=1}^{2^{s+1}} p_i = p([2^{s+1}]),$$

where in the last inequality we used the fact that there are at most $2^{s+1}$ positive instances in $\mathrm{MCSP}^A$. Now,

$$1 - \delta'(n) \leq p([2^{s+1}]) = p([2^{s+1}] \cap I_{c\beta}) + p([2^{s+1}] \setminus I_{c\beta}) \leq p(I_{c\beta}) + 2^{s+1} \cdot c\beta,$$

which implies that $p(I_{c\beta}) \geq 1 - 2^{s+1} \cdot c\beta - \delta'(n) \geq 1 - 2c\delta''(n) - \delta'(n) \geq 1 - 3c\delta''(n)$.

2. Note that an oracle $A$ can depend on input $x$, but $A$ must not depend on a specific $(f, t, s)$. Thus, we define $A$ so that, for *all* $(f, t, s) \in X$, the heavy queries $\{(T_1, s), \cdots, (T_{2^k}, s)\}$ (with respect to the distribution induced by $Q(x, r)$ where $r \sim R_{f,t,s}$) become positive instances. (Note that truth tables $T_i$ depend on $(f, t, s)$.) For any string $y$ of length at most $7 \log p(n)$, we define $A(y) := 1$ if and only if $A_0(y) = 1$, which ensures that $A$ and $A_0$ are consistent up to length $7 \log p(n)$. For each $(f, t, s) \in X$ such that $s > 7 \log p(n)$, define $k := s - 7 \log p(n)$; for each $i \in [2^k]$, we would like to define $A$ so that $I^A(d) = T_i$ for some description $d$ of length exactly equal to $s$. To this end, define $d := \langle 1^t, \langle f, s, \underline{i}_{k_i} \rangle \rangle$ and $A(\underline{j}_t, \langle f, s, \underline{i}_{k_i} \rangle) = T_{ij}$ for all $j \in [2^t]$, where $k_i$ is chosen so that $|d| = 2t + 2 \log s + k_i + O(1) = s$. Thus, $k_i := s - 2t - 2 \log s - O(1) \geq s - 7 \log p(n) = k$. This ensures that

$\underline{i}_{k_i}$ is well-defined. These imply that $\mathrm{K}_{I^A}(T_i) \leq s$; hence, $(T_i, s) \in \mathrm{MCSP}^A$ for all $i \in [2^k]$ and therefore $p([2^k]) \leq P_{\mathrm{id},t,s}$.

Now fix $(f, t, s) \in X$ such that $k = s - 7 \log p(n) > 0$ and $P_{\mathrm{id},t,s} \leq \delta'(n)$ hold. Since $k$ and $s$ are close, it holds that

$$\begin{aligned} & p([2^{s+1}]) \\ &= p(\{1, \cdots, 2^k\}) + p(\{2^k + 1, \cdots, 2 \cdot 2^k\}) + \cdots + p(\{2^{s+1} - 2^k + 1 \cdots, 2^{s+1}\}) \\ &\leq 2^{s+1}/2^k \cdot p([2^k]) \leq 2^{s+1-k} \cdot P_{\mathrm{id},t,s} = 2p(n)^7 \cdot P_{\mathrm{id},t,s} \leq 2p(n)^7 \cdot \delta'(n) = \delta''(n). \end{aligned}$$

We claim that this implies $p(I_\beta) \leq \delta''(n)$: Indeed, let $j := \max I_\beta$. If $j > 2^s$, then $\delta''(n) \geq p([2^{s+1}]) \geq p([2^{s+1}] \cap I_\beta) \geq \beta \cdot \min\{2^{s+1}, j\} > \beta \cdot 2^s = \delta''(n)$, which is a contradiction. Thus, we have $j \leq 2^s$, and hence $p(I_\beta) \leq p([2^{s+1}]) \leq \delta''(n)$ as desired.

□

This completes the proof of Lemma 8.13.

We conclude this chapter by posing an open question of extending Theorem 8.2 to the case of more-than-1-query reductions.

**Open Question 8.19.** Show that $\bigcap_A \mathsf{BPP}^{\mathrm{MCSP}^A[2]} \subseteq \mathsf{AM} \cap \mathsf{coAM}$.

# Chapter 9

# NP-hardness of MCSP for DNF-XOR Circuits

It was already shown by Masek [Mas79] in 1979 that it is NP-hard to solve MCSP for DNF formulas; here MCSP for DNF formulas is the problem of computing the minimum number of terms in a DNF formula consistent with a given truth table. In this chapter, we present the first progress about MCSP for such restricted circuit classes, by establishing an analogous result for the MCSP problem for depth-3 circuits of the form $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{XOR}$. Our techniques extend to an NP-hardness result for $\mathsf{MOD}_m$ gates at the bottom layer under inputs from $(\mathbb{Z}/m\mathbb{Z})^n$.

## 9.1 Introduction

As shown in Chapter 8, NP-hardness of MCSP is not likely to be resolved by a mere extension of current techniques, and thus it requires significantly new ideas. For now, given our inability to prove NP-hardness of MCSP, we should first try to consider a restricted version of MCSP, and develop more proof techniques towards resolving NP-hardness of the general MCSP.

### 9.1.1 MCSP for Restricted Circuit Classes

For a circuit class $\mathfrak{C}$, let $\mathfrak{C}$-MCSP denote MCSP for $\mathfrak{C}$; that is, $\mathfrak{C}$-MCSP asks for computing the minimum size of a $\mathfrak{C}$-circuit that computes a given truth table. Studying $\mathfrak{C}$-MCSP for restricted circuit classes $\mathfrak{C}$ is independently motivated by algorithmic applications in circuit minimization, learning theory (cf. [PV88, AHM$^+$08, Fel09, CIKK16]), and cryptography and lower bounds [RR97, BR17]. It was shown already in 1979 by Masek [Mas79] that DNF-MCSP is NP-hard. There have been different proofs of this result [Czo99, AHM$^+$08], and extensions to hardness of approximation [AHM$^+$08, Fel09, KS08]. Nevertheless, almost four decades after Masek's result, and despite the significant attention that the MCSP problem has received, NP-hardness of $\mathfrak{C}$-MCSP was not known for any natural class $\mathfrak{C}$ of circuits more expressive than DNFs.

As shown in Chapter 3, there is cryptographic evidence that MCSP is intractable; similarly, for a sufficiently expressive class $\mathfrak{C}$, some cryptographic evidence is known. For example, constant-depth threshold circuits and constant-depth Boolean circuits of large enough depth can compute some candidate pseudorandom function generator by a non-trivially small circuit, and thus $\mathfrak{C}$-MCSP for these circuit classes is not in polynomial time under some cryptographic assumptions (cf. [AHM$^+$08]). However, for classes extending DNFs that are not known to compute pseudorandom functions, no evidence of any sort for hardness

was known. To quote Allender et al. [AHM+08], "Thus an important open question is to resolve the NP-hardness of both learnability results as well as function minimization results above for classes that are stronger than DNF."

To summarize the current status of $\mathfrak{C}$-MCSP, there are two techniques of showing intractability of $\mathfrak{C}$-MCSP. One is based on cryptography, which shows an average-case hardness of $\mathfrak{C}$-MCSP for a sufficiently expressive class $\mathfrak{C}$ . Another is the result of Masek [Mas79], which shows that a worst-case complexity of DNF-MCSP is NP-hard. One may wonder whether the complexity of $\mathfrak{C}$-MCSP is monotone increasing with respect to $\mathfrak{C}$, and NP-hardness of DNF-MCSP implies NP-hardness of MCSP. This is indeed true in the case of *average-case complexity*: a natural property useful against $\mathfrak{C}'$ is also a natural property useful against $\mathfrak{C}$ for any classes $\mathfrak{C} \subseteq \mathfrak{C}'$. On the other hand, in the case of worst-case complexity, the same is not true: indeed, the limits of oracle-independent reductions (Theorem 8.1) show that MCSP $\not\leq_{\mathrm{T}}^{\mathsf{P}}$ MCSP$^A$ for some $A$ unless MCSP $\in \mathsf{P}$.

### 9.1.2 Our Results

The main result of this chapter is the first NP-hardness result for $\mathfrak{C}$-MCSP for a class $\mathfrak{C}$ of depth-3 circuits, namely the class of (unbounded fan-in) $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m$ circuits, where $m$ is any integer.

**Theorem 9.1.** *For every $m \geq 2$, given the truth table of a function $f\colon \mathbb{Z}_m^n \to \{0,1\}$, where $\mathbb{Z}_m = \mathbb{Z}/m\mathbb{Z} = \{0,1,\ldots,m-1\}$, it is NP-hard under polynomial-time deterministic many-one reductions to determine the size of the smallest $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m$ circuit $C$ that computes $f$, where circuit size is measured as the top fan-in of $C$.*

A few comments are in order. First, we elaborate on our computational model and complexity measure. We work with circuits which have an OR gate at the top, AND gates at the middle level, and $\mathsf{MOD}_m$ gates at the bottom level. We refer to such circuits as OR-AND-MOD circuits, or equivalently, DNF-MOD circuits. Such circuits operate in a natural way on inputs from $\mathbb{Z}_m^n$. We allow arbitrary constants from $\mathbb{Z}_m$ to feed in to gates at the bottom layer, and insist that inputs to the middle AND layer are Boolean. In other words, a $\mathsf{MOD}_m$ gate outputs 1 if and only if its corresponding linear equation over $\mathbb{Z}_m$ is satisfied, and the computations beyond the first layer are all Boolean. For $m = 2$, this is precisely the traditional model of DNF of Parities (cf. [CS16], [Juk06], [Juk12, Section 11.9], [ABG+14]).

The complexity measure we use is the top fan-in of the circuit, i.e., fan-in to the top OR gate. The main reason we work with this measure is naturalness and convenience. As argued in [CS16], top fan-in is the preferred measure for $\mathsf{OR\text{-}AND\text{-}MOD}_2$ circuits because: (*i*) it measures the number of affine subspaces required to cover the 1s of the function, and thus has a nice combinatorial meaning; (*ii*) the number of $\mathsf{MOD}_2$ gates feeding in to any middle layer AND gate can be assumed to be at most $n$ without loss of generality, by using basic linear algebra, and thus the top fan-in approximates the total number of gates to within a factor of $n$; and (*iii*) the size of a DNF is often measured by the number of terms in it, and analogously it makes sense to measure the size of a DNF of Parities by the top fan-in of the circuit.

Our results are not however critically dependent on the complexity measure we use, and admit different extensions. Indeed, we demonstrate the robustness of our techniques by adapting them to show a hardness result for computing the number of gates in $\mathsf{OR\text{-}AND\text{-}MOD}_p$ formulas, where $p$ is prime (Subsection 9.5.2).

Moreover, we mention that our approach can be modified to show a hardness of approximation result (Subsection 9.5.3).

### 9.1.3 Perspective: NP-hardness of MCSP for Other Circuit Classes

From Theorem 7.12, NP-hardness of $\mathfrak{C}$-MCSP under polynomial-time Turing reductions implies $2^{\Omega(n)}$ $\mathfrak{C}$-circuit lower bounds for $\mathsf{E}^{\mathsf{NP}}$ (or otherwise a fast deterministic simulation of nondeterministic algorithms follows, which is somewhat unexpected). In fact, it is easy to observe that our reduction yields a $2^{\Omega(n)}$ lower bound on the size of $\mathsf{DNF}\text{-}\mathsf{MOD}_2$ circuits for a function in $\mathsf{E}$, by applying our reduction on a trivial No instance of an $\mathsf{NP}$ language. Such strong exponential lower bounds for explicit functions have long been known for the model we consider (see e.g. [Gro98]). On the other hand, extending the $\mathsf{NP}$-hardness result even to slightly different classes such as depth-3 $\mathsf{AC}^0$ circuits might be a challenge, since it is still unknown if $\mathsf{E}^{\mathsf{NP}}$ requires depth-3 $\mathsf{AC}^0$ circuits of size $2^{\Omega(n)}$.

What might be more feasible though is showing $\mathsf{NP}$-hardness of $\mathfrak{C}$-MCSP for other related classes $\mathfrak{C}$ of circuits, and under weaker kinds of reductions, such as quasi-polynomial-time reductions or nonuniform reductions. For instance, it might be possible to extend our techniques to classes such as $\mathsf{THR} \circ \mathsf{AND} \circ \mathsf{MOD}$ and depth-3 $\mathsf{AC}^0$ circuits of small bottom fan-in. In these cases, exponential lower bounds of the form $2^{\Omega(n)}$ have been obtained (cf. [Gro98], [PSZ00]).

**Open Question 9.2.** Show that $\mathsf{NP}$-hardness of MCSP for $\mathsf{THR} \circ \mathsf{AND} \circ \mathsf{MOD}$ or depth-3 $\mathsf{AC}^0$ under polynomial-time Turing reductions.

Regarding weaker reducibility notions, there is a tradeoff between the running time of a reduction and the circuit lower bound in the barrier of Theorem 7.12; for example, it is not hard to see that $\mathsf{NP}$-hardness of $\mathfrak{C}$-MCSP under quasi-polynomial-time reductions is related to a $2^{n^\epsilon}$ circuit lower bound for some constant $\epsilon > 0$. Thus we conjecture that the following is a feasible and interesting open question, given the fact that $2^{\Omega(n^{1/(d-1)})}$ lower bounds have long been known for depth-$d$ $\mathsf{AC}^0$ circuits [Hås86].

**Open Question 9.3.** Show that $\mathsf{NP}$-hardness of $\mathsf{AC}^0_d$-MCSP under quasi-polynomial-time reductions or nonuniform reductions for some depth $d \geq 3$

More broadly, we believe that showing $\mathsf{NP}$-hardness of MCSP for more expressive classes $\mathfrak{C}$ is an important direction in better *understanding* circuit classes from the perspective of *meta-complexity*, i.e., complexity questions about computational problems involving circuits and algorithms. There are various criteria for measuring our understanding of a circuit class, for example, (*i*) Can we design non-trivial satisfiability algorithms for circuits in the class? (*ii*) Can we unconditionally construct pseudorandom generators secure against circuits in the class? (*iii*) Can we learn the class using membership queries under the uniform distribution? (*iv*) Can we prove lower bounds against proof systems whose lines are encoded by circuits in the class? We suggest that the $\mathsf{NP}$-hardness of $\mathfrak{C}$-MCSP is another strong indication that we understand a circuit class $\mathfrak{C}$ well.

### 9.1.4 Proof Overview

The rest of this chapter is dedicated to the proof of Theorem 9.1, which will be completed in Section 9.4. Here we provide a high-level description of the reduction. For simplicity, our exposition mostly focus on the case $m = 2$. After

that, we explain the main difficulties in extending the result to general $m$, and how these are addressed in our proof.

As mentioned above, Masek [Mas79] was the first to establish the NP-hardness of DNF minimization, and Theorem 9.1 can be interpreted as an extension of Masek's result to the more expressive DNF-MOD circuits. The structure of our argument follows however a *two-step* reduction introduced by Gimpel (cf. Allender et al. [AHM$^+$08]), brought to our attention thanks to an alternative proof of Masek's result from [AHM$^+$08]. More precisely, their work presents a new proof of the first stage of Gimpel's reduction, and provides a self-contained exposition of the entire argument.

Our NP-hardness proof for DNF-MOD circuits heavily builds on ideas of Gimpel and [AHM$^+$08], but the extension to depth-3 requires new ideas and makes the argument much more involved. Let ($\mathsf{DNF} \circ \mathsf{XOR}$)-MCSP be the computational problem described in Theorem 9.1 when $m = 2$, and let ($\mathsf{DNF} \circ \mathsf{XOR}$)-MCSP$^*$ be its natural generalization to *partial* boolean functions. In other words, an input to ($\mathsf{DNF} \circ \mathsf{XOR}$)-MCSP$^*$ encodes the truth table of a function $f\colon \{0,1\}^n \to \{0,1,*\}$, and we are interested in the size of the minimum ($\mathsf{DNF} \circ \mathsf{XOR}$)-circuit that agrees with $f$ on $f^{-1}(\{0,1\})$. Let $r \in \mathbb{N}$ be a large enough constant. Our proof reduces from the NP-complete problem $r$-Bounded Set Cover: Given a set system $\mathcal{S} \subseteq \binom{n}{\leq r}$ that covers $[n]$, determine the minimum number $\ell$ of sets $S_1, \ldots, S_\ell \in \mathcal{S}$ such that $\bigcup_{i=1}^{\ell} S_i = [n]$. (We refer to Subsection 9.2.3 for a precise formulation of these computational problems.)

In a bit more detail, we present a *randomized* (2-approximate) reduction from $r$-Bounded Set Cover to ($\mathsf{DNF} \circ \mathsf{XOR}$)-MCSP$^*$, and a *randomized* reduction from ($\mathsf{DNF} \circ \mathsf{XOR}$)-MCSP$^*$ to ($\mathsf{DNF} \circ \mathsf{XOR}$)-MCSP. These reductions are then efficiently derandomized using an appropriate pseudorandom generator. As opposed to previous works on the NP-hardness of DNF minimization, our proof crucially explores the fact that $r$-Bounded Set Cover is NP-hard even to *approximate* (by roughly a $\ln r$-factor), a result from [Fei98, Tre01b] (see Theorem 9.7, Subsection 9.2.3).

We discuss each reduction in more detail now. Common to both of them is a convenient characterization of the sets $C^{-1}(1) \subseteq \{0,1\}^n$ of inputs that can be accepted by non-trivial $\mathsf{AND} \circ \mathsf{XOR}$ circuits $C$. If $m$ is prime, it is not hard to show that this is precisely the class of affine subspaces of $\{0,1\}^n$. Consequently, for a non-trivial partial function $f\colon \{0,1\}^n \to \{0,1,*\}$, its corresponding $\mathsf{DNF}_{\mathsf{XOR}}(f)$ complexity is exactly the minimum number $t$ of affine subspaces $A_1, \ldots, A_t \subseteq \{0,1\}^n$ such that $f^{-1}(1) \subseteq \bigcup_{i=1}^{t} A_i$ and $\bigcup_{i=1}^{t} A_i \subseteq f^{-1}(\{1,*\})$ (see Subsection 9.2.2). The analysis of our polynomial-time reductions, which will not be covered in this section, rely on this characterization in fundamental ways.

**Step 1.** *A randomized reduction from $r$-Bounded Set Cover to* ($\mathsf{DNF} \circ \mathsf{XOR}$)-MCSP$^*$ (Subsection 9.3.1).

Given a set-system $\mathcal{S} \subseteq \binom{n}{\leq r}$, we define a partial boolean function $f\colon \{0,1\}^t \to \{0,1,*\}$, where $t = O(r \log n)$. This function is *probabilistically* constructed as follows. First, we associate to each $i \in [n]$ a *random* vector $v^i \in \{0,1\}^t$. For $S \in \mathcal{S}$, let $v^S = \{v^i \mid i \in S\}$. Then, we let $f$ be 1 on each input $v^i$, 0 on inputs that are *not* in the linear span of $v^S$ for every $S \in \mathcal{S}$, and $*$ elsewhere.

Using this construction, we are able to show by a delicate analysis that if $t$ is sufficiently large, the following holds with high probability: if $\mathcal{S}$ admits a cover of size $K$, then $\mathsf{DNF}_{\mathsf{XOR}}(f) \leq K$; moreover, if $\mathsf{DNF}_{\mathsf{XOR}}(f) \leq K$, then $\mathcal{S}$ admits a cover of size $\leq 2K$. (We discuss the intuition for this claim in Subsection 9.3.1.)

This construction and the hardness of approximation result for $r$-Bounded Set Cover imply that $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP$^*$ is NP-hard under many-one randomized reductions.

**Step 2.** *A randomized reduction from* $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP$^*$ *to* $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP (Subsection 9.3.2).

Let $f\colon \{0,1\}^t \to \{0,1,*\}$ be an instance of $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP$^*$. We *probabilistically* construct from $f$ a related *total* function $g\colon \{0,1\}^t \times \{0,1\}^s \to \{0,1\}$, where $r = t + 2$ and $s = O(r + t)$. In more detail, we encode for each $x \in \{0,1\}^t$ its corresponding value $f(x) \in \{0,1,*\}$ as a *boolean function* $g_x$ on a hypercube $\{0,1\}^s$. For an input $x$ such that $f(x) \in \{0,1\}$, we let $g(x0^s) = g_x(0^s) = f(x)$, where $g_x(\cdot) = 0$ elsewhere. On the other hand, if $f(x) = *$, we pick a *random* linear subspace $L_x \subseteq \{0,1\}^s$ of dimension $r$, and we encode $f(x)$ as the characteristic function of $L_x$.

Again, a careful argument allows us to establish the following connection between the partial function $f$ and the total function $g$: with high probability over the choice of the random linear subspaces $(L_x)_{x \in f^{-1}(*)}$, $\mathsf{DNF}_{\mathsf{XOR}}(g) = \mathsf{DNF}_{\mathsf{XOR}}(f) + |f^{-1}(*)|$. (We discuss the intuition for this claim in Subsection 9.3.2.) Consequently, it follows from this and the previous reduction that $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP is NP-hard under many-one randomized reductions.

**Step 3.** *Efficient derandomization of the reductions* (Subsection 9.4.1).

It is possible to prove that the first reduction is always correct provided that the collection of random vectors $v^i$ is *nice* with respect to the set-system $\mathcal{S}$ (Definition 9.14). Similarly, we can prove that the second reduction is correct whenever the collection $(L_x)_{x \in f^{-1}(*)}$ of linear subspaces is *scattered* (Definition 9.20). It turns out that both conditions can be checked in polynomial time. This implies that the previously discussed reductions are in fact *zero-error* reductions. Consequently, if we can efficiently construct nice vectors and scattered families of linear subspaces, the reductions can be made deterministic.

In order to achieve this, we use in both cases a subtle derandomization argument that relies on (polynomial-time computable) $\varepsilon$-biased distributions [NN93]. Recall that such distributions can fool arbitrary linear tests. By a more careful analysis, it is also known that they fool $\mathsf{AND} \circ \mathsf{XOR}$ circuits. We do *not* describe an $\mathsf{AND} \circ \mathsf{XOR}$ circuit to check if a collection of vectors is nice, or to check if a collection of linear subspaces is scattered. Still, we are able to show that if $\varepsilon < 2^{-s}$ then some scattered collection of linear subspaces is encoded by a string in the support of an $\varepsilon$-biased distribution, and that the same holds with respect to a nice collection of vectors if $\varepsilon < 2^{-t}$. In particular, trying all possible seeds of an $\varepsilon$-biased generator produces the combinatorial and algebraic objects that are sufficient to derandomize our reductions. (We refer to Subsection 9.4.1 for more details.)

Overall, combining the (derandomized) reductions and using the hardness of approximation result for $r$-Bounded Set Cover mentioned above, it follows that $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP is NP-hard under many-one deterministic polynomial-time reductions.

**The argument for arbitrary $m \geq 2$.** Let $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP and $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$ be the corresponding computational problems with respect to an arbitrary $m \geq 2$. (Recall that the input boolean functions in this case are defined

over $\mathbb{Z}_m^n$.) As we explain next, additional difficulties are present for general $m$.

An immediate challenge is that it is no longer clear if the analogue characterization (via affine subspaces) of the class of subsets of $\mathbb{Z}_m^n$ accepted by non-trivial $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits holds, and this is crucially exploited when $m = 2$. The main issue is that, while in the latter case the result can be established by elementary techniques using that $\mathbb{Z}_2^n$ is a *vector space* over $\mathbb{Z}_2$, for an arbitrary $m$ the underlying structure might be just a *module*. Without a *basis*, the result is less clear.

Nevertheless, it is possible to prove that the analogue result for $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits hold (cf. Lemma 9.4). The alternative and more general argument relies on a property of double orthogonal complements in $\mathbb{Z}_m^n$ (Subsection 9.5.1), and we refer to Subsection 9.2.2 for more details. Armed with this characterization, the reductions discussed before can be adapted to arbitrary $m$. Finding the right generalization of each definition requires some work, but after that, the *randomized* reductions for $m = 2$ and arbitrary $m \geq 2$ can be presented in a unified and transparent way.

In order to conclude the proof of Theorem 9.1, we need to derandomize the new reductions. For $m = 2$, the argument was based on an efficient construction of $\varepsilon$-biased distributions supported over $\{0,1\}^n$, and the fact that such distributions are also able to fool $\mathsf{AND} \circ \mathsf{XOR}$ circuits over $\{0,1\}^n$. Without going into further details, we mention that for arbitrary $m$ it is sufficient to use a pseudorandom generator that fools $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits over $\mathbb{Z}_m^n$. However, a generator with *near-optimal* dependency on $n$ and $\varepsilon$ is needed if we are hoping to obtain a polynomial-time reduction. We were not able to find such a result in the literature.[1]

We show in Subsection 9.4.2 that, for every $m \geq 2$, there is an efficient pseudorandom generator $G_n \colon \{0,1\}^{O(\log n + \log 1/\varepsilon)} \to \mathbb{Z}_m^n$ that $\varepsilon$-fools $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits of arbitrary size. Our construction relies on the efficient $\varepsilon$-biased generators for $\mathbb{Z}_m^n$ from [AMN98], together with a proof of the following result: If $G$ is an $\varepsilon$-biased generator against $\mathbb{Z}_m^n$, then $G$ $(m\varepsilon)$-fools $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits. Again, we cannot rely on a adaptation of the similar claim for $m = 2$, which requires a basis. Our proof proceeds instead by a careful analysis of certain exponential sums encoding the behaviour of the circuit, and that can be used to connect the distinguishing probability to the guarantees offered by the $\varepsilon$-biased generator. We refer to Subsection 9.4.2 for more details.

## 9.2 Preliminaries

### 9.2.1 Some notions from group theory

Let $m \geq 2$ be a constant. Let $\mathbb{Z}_m := \mathbb{Z}/m\mathbb{Z}$ denote the integers modulo $m$, where all operations on elements in $\mathbb{Z}_m = \langle +, \{0, 1, \ldots, m-1\} \rangle$ are taken mod $m$. For any integer $t \geq 1$, we regard $\mathbb{Z}_m^t$ as an additive group with componentwise addition. A non-empty subset $H \subseteq \mathbb{Z}_m^t$ is called a *linear subspace* if $H$ is a subgroup, that is, $0 \in H$ and $x + y \in H$ for any $x, y \in H$. A subset $A \subseteq \mathbb{Z}_m^t$ is called an *affine subspace* if $A$ is a coset, that is, there exist $a \in \mathbb{Z}_m^t$ and a linear subspace $H \subseteq \mathbb{Z}_m^t$ such that $A = H + a := \{\, h + a \mid h \in H \,\}$.

We stress that $\mathbb{Z}_m^t$ gives rise to a *module* and not to a *vector space* when $m$ is a composite number; however, we borrow some standard notation; for example,

[1] Existing generators seem to generate *bits* only, or are restricted to prime modulus, or can handle larger classes of functions but are not efficient enough for our purposes. We refer to [GKM15] and the references therein for related results.

for a scalar $c \in \mathbb{Z}_m$ and a "vector" $v \in \mathbb{Z}_m^t$, let $cv$ denote the scalar multiplication. Let $\langle x, y \rangle := \sum_{i=1}^t x_i y_i \ \ (\in \mathbb{Z}_m)$ for any $x, y \in \mathbb{Z}_m^t$ and $t \in \mathbb{N}$.

### 9.2.2 Circuit Size Measure and Its Characterization

For any integer $m \geq 2$, an $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_m$ $(= \mathsf{DNF} \circ \mathsf{MOD}_m)$ circuit is a $\mathsf{DNF}$ formula whose terms are $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits. Here, a $\mathsf{MOD}_m$ gate is a Boolean function such that $\mathsf{MOD}_m(x) = 1$ if and only if $\sum_{i=1}^t x_i \mod m = 0$ on input $x \in \{0,1\}^t$. We extend the input $\{0,1\}^t$ of a $\mathsf{MOD}_m$ gate to the larger domain $\mathbb{Z}_m^t$ in a natural way: that is, we regard the bottom $\mathsf{MOD}_m$ gate as a function $\mathsf{MOD}_m \colon \mathbb{Z}_m^* \to \{0,1\}$ that outputs 1 if and only if the sum of its input elements is congruent to 0 mod $m$. In this way, we can regard a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit as computing a function $f \colon \mathbb{Z}_m^t \to \{0,1\}$. We allow multiple input wires and access to constant input bits in the circuit. Note that this allows for more general equations to be computed by a bottom-layer modular gate.

The size of a circuit is usually defined as the number of gates. However, for us it is important to define the size of a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit as the top fan-in of the circuit, or equivalently, the number of $\mathsf{AND} \circ \mathsf{MOD}_m$ terms. (Note that the same size measure was used in [CS16] in the case $m = 2$.) For a function $f \colon \mathbb{Z}_m^t \to \{0,1\}$, define $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$ as the minimum number of terms of a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit computing $f$, i.e., the fan-in of its $\mathsf{OR}$ gate.

An $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit $C$ is said to *accept* a set $X \subseteq \mathbb{Z}_m^t$ if $x \in X \Leftrightarrow C$ outputs 1 on input $x$, for every $x \in \mathbb{Z}_m^t$. There is a nice combinatorial characterization of the set of inputs that such circuits can accept.

**Lemma 9.4** (Characterization of the power of $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits)**.** *Let $X \subseteq \mathbb{Z}_m^t$ be a nonempty set. Then, an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit accepts $X$ if and only if $X$ is an affine subspace of $\mathbb{Z}_m^t$.*

This is a standard fact when $m$ is a prime (cf. [CS16] for $m = 2$), in which case $\mathbb{Z}_m^t$ is a vector space. The same characterization holds when $m \geq 2$ is an arbitrary composite number, as established below. The proof relies on the following fact about orthogonal complements in the more general context of modules.

**Fact 9.5** (Double orthogonal complement)**.** *Let $H \subseteq \mathbb{Z}_m^t$ be a linear subspace, and let $H^\perp := \{\, x \in \mathbb{Z}_m^t \mid \sum_{i=1}^t x_i y_i = 0$ for any $y \in H \,\}$ be its orthogonal complement. Then, $(H^\perp)^\perp = H$.*

For completeness, we include a proof of this result in Subsection 9.5.1. Assuming Fact 9.5, we proceed to a proof of Lemma 9.4.

*Proof of Lemma 9.4.* Let $x := (x_1, \ldots, x_t) \in \mathbb{Z}_m^t$ denote the input to the circuit.

Suppose that an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit $\bigwedge_{k=1}^K C_k$ accepts $X$, where each $C_k$ is a $\mathsf{MOD}_m$ gate. Each $\mathsf{MOD}_m$ gate $C_k$ in the circuit defines a linear equation over $(x_1, \ldots, x_t)$. That is, there are coefficients $a_k^1, \ldots, a_k^t \in \mathbb{Z}_m$ and an element $b_k \in \mathbb{Z}_m$ such that $\sum_{i=1}^t a_k^i x_i = b_k$ if and only if $C_k$ accepts the input $x$. Therefore, the circuit $\bigwedge_{k=1}^K C_k$ accepts the intersection of such linear equations over $\mathbb{Z}_m$. Specifically, for a matrix $A := (a_k^i)_{k \in [K], i \in [t]}$ and a vector $b := (b_k)_{k \in [K]}$, the circuit accepts all inputs $x \in \mathbb{Z}_m^t$ such that $Ax = b$; namely, $X = \{\, x \in \mathbb{Z}_m^t \mid Ax = b \,\}$. Since $X$ is nonempty, we can take some element $x_0 \in X$. Now, we can rewrite $X$ as

$$X = \{\, x \in \mathbb{Z}_m^t \mid A(x - x_0) = 0 \,\} = \{\, y \in \mathbb{Z}_m^t \mid Ay = 0 \,\} + x_0,$$

which is an affine subspace of $\mathbb{Z}_m^t$.

For the converse direction, we use the notion of orthogonal complement. Suppose that $X \subseteq \mathbb{Z}_m^t$ is an affine subspace. By definition, we can decompose $X$ into a linear subspace $H \subseteq \mathbb{Z}_m^t$ and a shift $a \in \mathbb{Z}_m^t$ so that $X = H + a$.

We first claim that $H$ can be accepted by some $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit. To prove this, it is sufficient to show the existence of some matrix $A \in \mathbb{Z}_m^{K \times t}$ such that $H = \{ x \in \mathbb{Z}_m^t \mid Ax = 0 \}$. Since $H$ is a linear subspace, by Fact 9.5, for any $x \in \mathbb{Z}_m^t$,

$$x \in H \quad \text{if and only if} \quad \sum_{i=1}^{t} x_i \cdot y_i = 0 \text{ for every } y \in H^\perp.$$

That is, we can define a matrix $A \in \mathbb{Z}_m^{|H^\perp| \times t}$ as $(y_i)_{y \in H^\perp, i \in [t]}$. (In other words, for each $y \in H^\perp$, we add a $\mathsf{MOD}_m$ gate that checks if $\sum_{i=1}^t x_i \cdot y_i = 0$, where each coefficient $y_i$ is simulated using multiple input wires.)

To accept $X$, we just need to shift $H$ by $a$. Indeed, for a vector $b := Aa$, we have $X = H + a = \{ x \in \mathbb{Z}_m^t \mid Ax = b \}$; thus we can construct an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit accepting $X$ by simulating the condition $Ax = b$. □

As a consequence of Lemma 9.4, for a function $f \colon \mathbb{Z}_m^t \to \{0, 1\}$, the minimum size of a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit computing $f$ equals the minimum number $S$ of affine subspaces $T_1, \ldots, T_S \subseteq \mathbb{Z}_m^t$ such that $\bigcup_{i=1}^S T_i = f^{-1}(1)$.

### 9.2.3 Computational Problems

We formulate computational problems as optimization problems for simplicity; however, one can easily see that the problems are equivalent to decision versions.

The starting point of our NP-hardness results is the set cover problem on instances where each set has size at most $r$.

**Definition 9.6** ($r$-Bounded Set Cover Problem)**.** *For an integer $r \in \mathbb{N}$, the $r$-Bounded Set Cover Problem is defined as follows:*

- Input. *An integer $n \in \mathbb{N}$ and a collection $\mathcal{S} \subseteq 2^{[n]}$ of nonempty subsets of the universe $[n]$ such that $|S| \leq r$ for each $S \in \mathcal{S}$, and $\bigcup_{S \in \mathcal{S}} S = [n]$.*
- Output. *The minimum number $\ell$ of subsets $S_1, \ldots, S_\ell \in \mathcal{S}$ such that $\bigcup_{i=1}^{\ell} S_i = [n]$.*

For this problem, a tight inapproximability result based on NP-hardness is known.

**Theorem 9.7** (Feige [Fei98], Trevisan [Tre01b])**.** *Let $r$ be a sufficiently large constant. It is NP-hard (under polynomial-time many-one reductions) to approximate the solution of the $r$-bounded set cover problem within a factor of $\ln r - O(\ln \ln r)$. That is, for any language $L \in \mathsf{NP}$, there exists a polynomial-time machine that, on input $x$, outputs a threshold $\theta$ and an instance $\mathcal{S}$ of the $r$-bounded set cover problem such that if $x \in L$ then $\mathcal{S}$ has a cover of size at most $\theta$, and if $x \notin L$ then $\mathcal{S}$ does not have a cover of size at most $\theta \cdot (\ln r - O(\ln \ln r))$.*

We stress that the *inapproximability* result is essential for us; we will present a reduction from a 2-factor approximation of the $r$-bounded set cover problem to the minimum $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit minimization problem.

**Definition 9.8** (Minimum Circuit Size Problem for $\mathsf{DNF}\circ\mathsf{MOD}_m$)**.** *For an integer $m \geq 2$, the Minimum Circuit Size Problem for $\mathsf{DNF} \circ \mathsf{MOD}_m$, abbreviated as $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP, is defined as follows:*

- Input. *A Boolean function $f\colon \mathbb{Z}_m^t \to \{0,1\}$, represented as a truth table of length $m^t$.*
- Output. $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$.

While our final theorem confirms that $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP is NP-hard, we will first prove NP-hardness of the circuit minimization problem on instances of a partial function $f : \mathbb{Z}_m^t \to \{0,1,*\}$. That is, we regard any input $x \in f^{-1}(*)$ as "undefined." For a partial function $f\colon \mathbb{Z}_m^t \to \{0,1,*\}$, we say that a circuit $C$ computes $f$ if $C(x) = f(x)$ for any $x \in f^{-1}(\{0,1\})$. We extend the definition of $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$ to the size of the minimum $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit computing the partial function $f\colon \mathbb{Z}_m^t \to \{0,1,*\}$. The following problem is concerned with the circuit size of partial functions, and we distinguish it from the problem above by adding a superscript $*$.

**Definition 9.9** (Minimum Circuit Size Problem for Partial Functions)**.** *For an integer $m \geq 2$, the Minimum Circuit Size Problem$^*$ for $\mathsf{DNF}\circ\mathsf{MOD}_m$, abbreviated as $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$, is defined as follows:*

- Input. *A Boolean function $f\colon \mathbb{Z}_m^t \to \{0,1,*\}$, represented as a string of length $m^t$ over the alphabet $\{0,1,*\}$.*
- Output. $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$.

## 9.3 Hardness of $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP Under Randomized Reductions

### 9.3.1 Reduction from $r$-Bounded Set Cover to $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$

This subsection is devoted to proving the following theorem.

**Theorem 9.10.** $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP$^*$ *is* NP*-hard under (zero-error) randomized polynomial-time many-one reductions.*

Let $r$ be a large enough constant so that the approximation factor of $\ln r - O(\ln \ln r)$ in Theorem 9.7 is larger than 2. We present a reduction from a 2-factor approximation of the $r$-bounded set cover problem to $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$.

Let us prepare some notation. Let $\mathcal{S}$ be an instance of the $r$-bounded set cover problem over the universe $[n]$ (in particular, $\bigcup_{S\in\mathcal{S}} S = [n]$). Let $t \in \mathbb{N}$ be a parameter chosen later. For each $i \in [n]$, pick $v^i \sim \mathbb{Z}_m^t$ independently and uniformly at random. For any $S \subseteq [n]$, let $v^S$ denote $\{\, v^i \mid i \in S \,\}$. Let $\mathsf{span}(v^S) := \{\, \sum_{i\in S} c_i \cdot v^i \mid c_i \in \mathbb{Z}_m \text{ for any } i \in S \,\}$ denote the linear span of $v^S$. (Note that $\mathsf{span}(v^S)$ is a linear subspace of $\mathbb{Z}_m^t$ whenever $S \neq \varnothing$.) In our reduction, an element $i \in [n]$ is mapped to a random point $v^i$ of $\mathbb{Z}_m^t$, and a set $S \in \mathcal{S}$ corresponds to a linear subspace $\mathsf{span}(v^S)$.

For any set cover instance $\mathcal{S}$, we define a function $f : \mathbb{Z}_m^t \to \{0,1,*\}$ as

$$f(x) := \begin{cases} 1 & (\text{if } x = v^i \text{ for some } i \in [n]) \\ 0 & (\text{if } x \notin \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)) \\ * & (\text{otherwise}) \end{cases}$$

for any $x \in \mathbb{Z}_m^t$. The truth table of $f$ is the output of our reduction.

It is not hard to see that $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$ is at most the minimum set cover size for $\mathcal{S}$ (Claim 9.11 below). Of course, the difficulty is in proving a circuit lower bound for $f$ (Claim 9.12 below).

The idea is as follows: For simplicity of the exposition, let us focus on the case of $m = 2$, and moreover let us first consider the case of a $\mathsf{DNF} \circ \mathsf{MOD}_2$ circuit $C$ for $f$ that accepts a union of *linear* subspaces (instead of affine subspaces). More precisely, let $C^{-1}(1)$ be a union of linear subspaces $\{T_k\}_{k\in[K]}$. Then $T_k$ is a subset of $C^{-1}(1) \subseteq f^{-1}(\{1, *\}) = \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)$; furthermore, each $\mathsf{span}(v^S)$ is a random linear subspace of small dimension $r$; therefore, it is possible to show that, with high probability, the set $\{\, i \in [n] \mid v^i \in T_k \,\}$ of points covered by $T_k$ is contained in some legal set $S \in \mathcal{S}$ of the set cover instance; hence the circuit size $K$ is at least the minimum set cover size.

In the case that a circuit $C$ accepts the union of *affine* subspaces, it is no longer true that, for any affine subspace $T$ such that $T \subseteq \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)$, the set $\{\, i \in [n] \mid v^i \in T \,\}$ is covered by some legal set $S \in \mathcal{S}$; indeed, for any two points $v^i$ and $v^j$, the set $\{v^i, v^j\}$ $\big(= v^i \oplus \{0, v^i \oplus v^j\}\big)$ is an affine subspace of $\mathbb{Z}_2^t$, whereas $\{i, j\}$ is not necessarily legal in the set cover instance $\mathcal{S}$. Nonetheless, we can still prove that, with high probability, the set $\{\, i \in [n] \mid v^i \in T \,\}$ is covered by two legal sets $S_1, S_2 \in \mathcal{S}$. As a consequence, the minimum number of affine subspaces needed to cover $v^1, \ldots, v^n$ gives us a 2-factor approximation of the minimum set cover size for $\mathcal{S}$. By Theorem 9.7, it follows that $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP$^*$ is NP-hard under randomized reductions. Details follow.

**Claim 9.11** (Easy part)**.** *Suppose that $\mathcal{S}$ has a set cover of size $K$. Then $\mathsf{DNF}_{\mathsf{MOD}_m}(f) \le K$.*

*Proof.* Let $\mathcal{C} \subseteq \mathcal{S}$ be a set cover of size $K$. For each $S \in \mathcal{C}$, by Lemma 9.4, there exists an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit $C_S$ such that $C_S$ accepts $\mathsf{span}(v^S)$. Define a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit $C := \bigvee_{S\in\mathcal{C}} C_S$. It is easy to see that $C$ computes $f$. □

Conversely, we prove the following:

**Claim 9.12** (Hard part)**.** *For some parameter $t$ such that $m^t = (nm)^{O(r)}$, the following holds with probability at least $\frac{1}{2}$ (over the choice of $(v^i)_{i\in[n]}$):*

*Let $K := \mathsf{DNF}_{\mathsf{MOD}_m}(f)$. Then $\mathcal{S}$ has a set cover of size $2K$.*

The two claims above imply that $2 \cdot \mathsf{DNF}_{\mathsf{MOD}_m}(f)$ is a 2-factor approximation for the set cover problem: indeed, let $s$ be the minimum set cover size for $\mathcal{S}$; then we have $s \le 2 \cdot \mathsf{DNF}_{\mathsf{MOD}_m}(f) \le 2s$. It thus remains to prove Claim 9.12.

To prove Claim 9.12, let us clarify the desired condition that random objects $(v^i)_{i\in[n]}$ should satisfy. For any $I \subseteq [n]$, define the *affine span* of $v^I$ as

$$\mathsf{affine\text{-}span}(v^I) := \Big\{ \sum_{i\in I} c_i v^i \mid c_i \in \mathbb{Z}_m \text{ for } i \in I \text{ and } \sum_{i\in I} c_i = 1 \Big\}.$$

The important property of the affine span is that, if an affine subspace $A$ covers the set $v^I$ of points in $I \subseteq [n]$, then its affine span must also be covered by $A$.

**Claim 9.13** (Property of the affine span)**.** *For any affine subspace $A$ of $\mathbb{Z}_m^t$ and any $I \subseteq [n]$, if $v^I \subseteq A$ then $\mathsf{affine\text{-}span}(v^I) \subseteq A$.*

*Proof.* Let us write $A = H + a$ for some linear space $H \subseteq \mathbb{Z}_m^t$ and vector $a \in \mathbb{Z}_m^t$. Since $v^i \in v^I \subseteq A$ for each $i \in I$, there exists some vector $h^i \in H$ such that

$v^i = h^i + a$. Take any coefficients $(c_i)_{i \in I}$ such that $c_i \in \mathbb{Z}_m$ and $\sum_{i \in I} c_i = 1$. Then,

$$\sum_{i \in I} c_i v^i = \sum_{i \in I} c_i (h^i + a) = \sum_{i \in I} c_i h^i + a \ \in \ H + a.$$

□

By Lemma 9.4, the circuit size of $f$ equals the minimum number of affine subspaces $A_1, \ldots, A_K \subseteq f^{-1}(\{1, *\})$ such that $\bigcup_{i=1}^{K} A_i \supseteq f^{-1}(1)$. Intuitively, we would like to require that, if the set $v^I \ \big(\subseteq f^{-1}(1)\big)$ of points is covered by some affine subspace $A \subseteq f^{-1}(\{1, *\})$, then there exist two legal sets $S_1, S_2$ of the set cover instance $\mathcal{S}$ such that $I \subseteq S_1 \cup S_2$. In fact, one of these sets can be taken as a singleton:

**Definition 9.14.** *We say that* $(v^i)_{i \in [n]}$ *is* nice *(with respect to* $\mathcal{S}$*) if, for any* $I \subseteq [n]$*,*

$$\mathsf{affine\text{-}span}(v^I) \subseteq \bigcup_{S \in \mathcal{S}} \mathsf{span}(v^S) \quad \Longrightarrow \quad I \subseteq S_I \cup \{i_I\} \tag{9.1}$$

*for some* $S_I \in \mathcal{S}$ *and* $i_I \in [n]$*.*

We will prove that $(v^i)_{i \in [n]}$ is nice with probability at least $\frac{1}{2}$, and that for any nice $(v^i)_{i \in [n]}$, the minimum size of $\mathsf{DNF} \circ \mathsf{MOD}_m$ is a 2-factor approximation of the minimum set cover size. We prove the latter first:

**Claim 9.15.** *Let* $(v^i)_{i \in [n]}$ *be nice, and* $K := \mathsf{DNF}_{\mathsf{MOD}_m}(f)$*. Then* $\mathcal{S}$ *has a set cover of size* $2K$*.*

*Proof.* Let $C = \bigvee_{k=1}^{K} C_k$ be a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit computing $f$, where each $C_k \in \mathsf{AND} \circ \mathsf{MOD}_m$ is nontrivial. By Lemma 9.4, $C_k^{-1}(1)$ is an affine subspace of $\mathbb{Z}_m^t$. For each $C_k$, we will choose 2 sets from $\mathcal{S}$ so that the union of all these sets cover the universe $[n]$.

Fix any $C_k$ and let $I_k := \{\, i \in [n] \mid C_k(v^i) = 1 \,\}$ be the set of all points covered by $C_k$. Since $C_k^{-1}(1)$ is an affine subspace of $\mathbb{Z}_m^t$ and $v^{I_k} \subseteq C_k^{-1}(1)$, we have $\mathsf{affine\text{-}span}(v^{I_k}) \subseteq C_k^{-1}(1)$ by Claim 9.13. Since the circuit $C$ computes $f$, $C_k^{-1}(1) \subseteq C^{-1}(1) \subseteq f^{-1}(\{1, *\}) = \bigcup_{S \in \mathcal{S}} \mathsf{span}(v^S)$. Thus we have $\mathsf{affine\text{-}span}(v^{I_k}) \subseteq \bigcup_{S \in \mathcal{S}} \mathsf{span}(v^S)$, which means that the hypothesis of niceness (9.1) is satisfied; hence there exist some subset $S_{k1} \in \mathcal{S}$ and some element $i_k \in [n]$ such that $I_k \subseteq S_{k1} \cup \{i_k\}$. Take any set $S_{k2} \in \mathcal{S}$ such that $i_k \in S_{k2}$ (such a set $S_{k2}$ must exist because we assumed $\bigcup_{S \in \mathcal{S}} S = [n]$). Then $I_k \subseteq S_{k1} \cup S_{k2}$.

Now we claim that $\bigcup_{k=1}^{K} S_{k1} \cup S_{k2} = [n]$ (and hence the set cover instance $\mathcal{S}$ has a cover of size $2K$). Indeed, for any $i \in [n]$, we have $f(v^i) = 1$ and hence $C(v^i) = 1$, which means that there exists some subcircuit $C_k$ such that $C_k(v^i) = 1$. Thus $i \in I_k \subseteq S_{k1} \cup S_{k2}$ for some $k \in [K]$. □

It remains to show that a random choice of $(v_i)_{i \in [n]}$ is nice with high probability:

**Claim 9.16.** *For each* $i \in [n]$*, pick* $v^i \sim \mathbb{Z}_m^t$ *uniformly at random and independently. If* $t \geq r + ((r+2) \log n + \log |\mathcal{S}| + 1) / \log m$*, then* $(v^i)_{i \in [n]}$ *is nice with probability at least* $\frac{1}{2}$*.*

To prove Claim 9.16, we will use a union bound over all relevant subsets $I \subseteq [n]$; however, the definition of niceness (9.1) appears to suggest that we need to take a union bound over exponentially many subsets $I$. The next claim shows that this is in fact *not* the case.

**Claim 9.17** (Characterization of niceness)**.** *$(v^i)_{i\in[n]}$ is not nice (with respect to $\mathcal{S}$) if and only if there exists some subset $I \subseteq [n]$ such that all the following conditions hold:*

1. *$|I| \le r+2$,*
2. *$I \not\subseteq S \cup \{i\}$ for any $S \in \mathcal{S}$ and $i \in [n]$, and*
3. *$\mathsf{affine\text{-}span}(v^I) \subseteq \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)$.*

In particular, there are at most $n^{r+2}$ subsets $I \subseteq [n]$ over which we need to take a union bound.

*Proof.* By the definition of niceness, $(v^i)_{i\in[n]}$ is not nice if and only if there exists some subset $I \subseteq [n]$ such that $\mathsf{affine\text{-}span}(v^I) \subseteq \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)$ whereas $I \not\subseteq S \cup \{i\}$ for any $S \in \mathcal{S}$ and $i \in [n]$. Therefore, it is clear that the three conditions imply that $(v^i)_{i\in[n]}$ is not nice; we prove below the converse direction (the "only if" part of Claim 9.17).

A crucial observation is that, for any subset $I \subseteq [n]$ of size at least $r+2$, the second condition always holds: Indeed, recall that $\mathcal{S}$ is an instance of the $r$-bounded set cover instance; that is, $|S| \le r$ for any $S \in \mathcal{S}$. Hence, for any $S \in \mathcal{S}$ and $i \in [n]$, we have $|S \cup \{i\}| \le r+1$; thus $I$ cannot be a subset of $S \cup \{i\}$ simply because $|I| \ge r+2$.

Now suppose that there exists some subset $I \subseteq [n]$ satisfying the second and third conditions, but not the first one, that is, $|I| > r+2$. Take any subset $I' \subseteq I$ such that $|I'| = r+2$. We claim that $I'$ satisfies all three conditions: The first condition ($|I'| \le r+2$) is obvious. The second condition holds because of the observation above. To see the third condition, by assumption, we have $\mathsf{affine\text{-}span}(v^I) \subseteq \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)$; hence, we also have $\mathsf{affine\text{-}span}(v^{I'}) \subseteq \mathsf{affine\text{-}span}(v^I) \subseteq \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)$. □

Now let us proceed to a proof of Claim 9.16.

*Proof of Claim 9.16.* We will bound the probability that a random $(v^i)_{i\in[n]}$ is not nice, by using the union bound over all the subsets $I \subseteq [n]$ such that the first and second conditions in Claim 9.17 hold. To this end, fix any subset $I \subseteq [n]$ such that $|I| \le r+2$ and $I \not\subseteq S \cup \{i\}$ for any $S \in \mathcal{S}$ and $i \in [n]$ (in particular, $I$ is not empty). We would like to bound the probability that the affine subspace of $v^I$ is a subset of $\bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)$.

Take an arbitrary (*e.g.* the smallest) element $i_0 \in I$. Define coefficients $(c_i)_{i\in I}$ as follows: $c_i := 1 \in \mathbb{Z}_m$ for any $i \in I \setminus i_0$ and $c_{i_0} := (2 - |I|) \mod m \in \mathbb{Z}_m$. By this definition, we have $\sum_{i\in I} c_i = 1$; hence, $\sum_{i\in I} c_i v^i \in \mathsf{affine\text{-}span}(v^I)$. Therefore,

$$\begin{aligned}\Pr_{v^1,\ldots,v^n}\left[\mathsf{affine\text{-}span}(v^I) \subseteq \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)\right] &\le \Pr\left[\sum_{i\in I} c_i v^i \in \bigcup_{S\in\mathcal{S}} \mathsf{span}(v^S)\right] \\ &\le \sum_{S\in\mathcal{S}} \Pr\left[\sum_{i\in I} c_i v^i \in \mathsf{span}(v^S)\right].\end{aligned}$$

By the assumption on $I$, we have $I \not\subseteq S \cup \{i_0\}$ for any $S \in \mathcal{S}$; that is, there exists some index $j_S \in I \setminus \{i_0\} \setminus S$. Note that $c_{j_S} = 1$ because $j_S \in I \setminus \{i_0\}$.

Therefore, the last probability is

$$\begin{aligned}
\sum_{S\in\mathcal{S}} \Pr\left[\sum_{i\in I} c_i v^i \in \mathsf{span}(v^S)\right] &= \sum_{S\in\mathcal{S}} \Pr\left[v^{j_S} \in \mathsf{span}(v^S) - \sum_{i\in I\setminus\{j_S\}} c_i v^i\right] \\
&= \sum_{S\in\mathcal{S}} \Pr\left[v^{j_S} = \sum_{i\in S} d_i v^i - \sum_{i\in I\setminus\{j_S\}} c_i v^i \text{ for some } (d_i)_{i\in S}\right] \\
&= \sum_{S\in\mathcal{S}} \sum_{(d_i)_{i\in S}} \Pr\left[v^{j_S} = \sum_{i\in S} d_i v^i - \sum_{i\in I\setminus\{j_S\}} c_i v^i\right] \\
&\le |\mathcal{S}| \cdot m^r \cdot m^{-t},
\end{aligned}$$

where the last inequality holds because the random vector $v^{j_S}$ does not appear in the right summations.

Finally, by taking the union bound over all $I$ such that $|I| \le r+2$ (and $I \not\subseteq S \cup \{i\}$ for any $S \in \mathcal{S}$ and $i \in [n]$), the probability that $(v^i)_{i\in[n]}$ is not nice is bounded from above by $n^{r+2} \cdot |\mathcal{S}| \cdot m^{r-t} \le \frac{1}{2}$. □

Given these claims above, it is immediate to complete the whole proof.

*Proof of Claim 9.12.* We may assume without loss of generality that $|\mathcal{S}| \le n^r$ since $\mathcal{S}$ is an instance of the $r$-bounded set cover problem. We set $t \in \mathbb{N}$ to be the smallest integer such that $t \ge r + ((r+2)\log n + \log|\mathcal{S}| + 1)/\log m$; then $t = O(r\log(nm)/\log m)$. (Here the $O$ notation hides only a universal constant.) Combining Claim 9.15 and 9.16, we immediately obtain Claim 9.12. □

*Proof of Theorem 9.10.* The encoding of the function $f\colon \mathbb{Z}_m^t \to \{0,1,*\}$ is of size $O(m^t) = (nm)^{O(r)}$, which is a polynomial in the input size $\mathsf{poly}(n, |\mathcal{S}|)$.

Moreover, it is possible to make the reduction zero-error: Indeed, the condition of the niceness can be checked in polynomial time, by using the characterization of Claim 9.17.

Finally, recall that the $r$-bounded set cover problem is NP-hard to approximate within a factor of 2 by Theorem 9.7 for a sufficiently large constant $r \in \mathbb{N}$. Hence, NP-hardness of $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP$^*$ follows from Claim 9.11 and 9.12. □

### 9.3.2 Reduction from $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP$^*$ to $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP

Next, we present a reduction for the minimum circuit size problem for partial functions to that for total functions:

**Theorem 9.18.** *There is a (zero-error) randomized polynomial-time many-one reduction from* $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP$^*$ *to* $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP*.*

Let $f\colon \mathbb{Z}_m^t \to \{0,1,*\}$ be an instance of $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP$^*$. Let $r := t+2$ and $s := \lceil (2r+2t)\log m + 2\rceil = \lceil 4(t+1)\log m + 2\rceil$. We encode each value $f(x) \in \{0,1,*\}$ of the partial function $f$ as a function on a "hypercube" $\mathbb{Z}_m^s$: namely, we construct a new *total* function $g : \mathbb{Z}_m^t \times \mathbb{Z}_m^s \to \{0,1\}$ such that $f(x)$ corresponds to $(g(x,y))_{y\in\mathbb{Z}_m^s}$. Specifically, if $f(x) \ne *$, then $f(x)$ is encoded as a hypercube whose origin[2] $0^s$ is assigned $f(x)$ and other points are assigned 0; if

[2] $0^s$ denotes the zero of $\mathbb{Z}_m^s$ for any $s \in \mathbb{N}$.

$f(x) = *$, then we pick a random linear subspace $L_x \subseteq \mathbb{Z}_m^s$ of dimension $r$ and we encode $f(x)$ as the characteristic function of $L_x$.

Formally, for each $x \in f^{-1}(*)$, we pick $v_x^1, \dots, v_x^r \sim \mathbb{Z}_m^s$ uniformly and independently at random, and define a random linear subspace $L_x := \mathsf{span}(v_x^1, \dots, v_x^r)$. Then the output $g : \mathbb{Z}_m^t \times \mathbb{Z}_m^s \to \{0,1\}$ of our reduction is defined as

$$g(x,y) := \begin{cases} f(x) & (\text{if } f(x) \in \{0,1\} \text{ and } y = 0^s) \\ 1 & (\text{if } f(x) = * \text{ and } y \in L_x) \\ 0 & (\text{otherwise}) \end{cases}$$

for any $(x,y) \in \mathbb{Z}_m^t \times \mathbb{Z}_m^s$.

The idea is as follows: Let us imagine how a minimum $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit $C$ computing $g$ looks like. We need to cover $g^{-1}(1)$ by as few affine subspaces as possible. Note that $g^{-1}(1)$ consists of two parts: $\{(x, 0^s)\}$ for each $x \in f^{-1}(1)$, and $\{x\} \times L_x$ for each $x \in f^{-1}(*)$. In order to cover the latter one, it is likely that we need to use the affine subspace $\{x\} \times L_x$ itself for each $x \in f^{-1}(*)$; indeed, since each $L_x$ is a random linear subspace, under our constraints with high probability there is no affine subspace which simultaneously covers (a large fraction of) two random affine subspaces $\{x\} \times L_x$ and $\{x'\} \times L_{x'}$ for $x \neq x' \in f^{-1}(*)$ (Claim 9.23 below). Therefore, the minimum circuit $C$ should contain a subcircuit which accepts $\{x\} \times L_x$ for each $x \in f^{-1}(*)$. Now it remains to cover $\{(x, 0^s)\}$ for each $x \in f^{-1}(1)$, but here we can *optionally* cover $\{(x, 0^s)\}$ for each $x \in f^{-1}(*)$ (which has been already covered by $\{x\} \times L_x$). This is exactly the same situation as $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$; thus with high probability we have $\mathsf{DNF}_{\mathsf{MOD}_m}(g) = \mathsf{DNF}_{\mathsf{MOD}_m}(f) + |f^{-1}(*)|$. Details follow.

**Claim 9.19.** $\mathsf{DNF}_{\mathsf{MOD}_m}(g) \leq \mathsf{DNF}_{\mathsf{MOD}_m}(f) + |f^{-1}(*)|$.

*Proof.* Suppose that a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit $C = \bigvee_{k=1}^K C_k$ computes $f$. For each $x^* \in f^{-1}(*)$, take an $\mathsf{AND} \circ \mathsf{MOD}_m$ formula $C_{x^*}$ such that $C_{x^*}^{-1}(1) = \{x^*\} \times L_{x^*}$ (by Lemma 9.4). Define $C'(x,y) := \bigvee_{k=1}^K (C_k(x) \wedge (y_1 = 0) \wedge \cdots \wedge (y_s = 0)) \vee \bigvee_{x^* \in f^{-1}(*)} C_{x^*}(x,y)$. It is easy to see that $C'(x,y) = g(x,y)$ for any $(x,y) \in \mathbb{Z}_m^t \times \mathbb{Z}_m^s$. □

In order to prove the other direction, let us clarify the desired condition for random linear spaces. We require that $(L_x)_{x \in f^{-1}(*)}$ is pairwise "disjoint" and that each $L_x$ is nondegenerated.

**Definition 9.20.** *We say that* $(L_x)_{x \in f^{-1}(*)}$ *is* scattered *if* $|L_x| = m^r$ *and* $L_x \cap L_{x'} = \{0^s\}$ *for any distinct* $x, x' \in f^{-1}(*)$.

It is easy to prove that the collection of random linear spaces satisfies the condition above.

**Claim 9.21.** $(L_x)_{x \in f^{-1}(*)}$ *is scattered with probability at least* $\frac{1}{2}$*, provided that* $s \geq (2r + 2t) \log m + 2$.

*Proof.* We first bound the probability that $(L_x)_{x \in f^{-1}(*)}$ is not pairwise disjoint.

$$\begin{aligned}
&\Pr\left[\ L_x \cap L_{x'} \neq \{0^s\} \text{ for some distinct } x, x' \in f^{-1}(*)\ \right] \\
&\leq \sum_{x \neq x' \in f^{-1}(*)} \Pr\left[L_x \cap L_{x'} \neq \{0^s\}\right] \\
&\leq \sum_{x \neq x' \in f^{-1}(*)} \Pr\left[\sum_{i=1}^{r} c_i v_x^i = \sum_{i=1}^{r} d_i v_{x'}^i \text{ for some nonzero } (c_i)_{i\in[r]}, (d_i)_{i\in[r]}\right] \\
&< m^{2t} \cdot m^{2r} \cdot 2^{-s} \ \leq\ \frac{1}{4},
\end{aligned}$$

where, in the last line, we used the fact that the probability that $\sum_{i=1}^{r} c_i v_x^i = \sum_{i=1}^{r} d_i v_{x'}^i$ is at most $2^{-s}$ for nonzero (*i.e.* $c_i \neq 0, d_j \neq 0$ for some $i, j \in [r]$) coefficients $(c_i)_{i\in[r]}, (d_i)_{i\in[r]}$.[3]

Next, we bound the probability that $|L_x| < m^r$. Indeed,

$$\begin{aligned}
&\Pr\left[\ |L_x| < m^r \text{ for some } x \in f^{-1}(*)\ \right] \\
&\leq \sum_{x \in f^{-1}(*)} \Pr\left[\sum_{i=1}^{r} c_i v_x^i = 0^s \text{ for some nonzero } (c_i)_{i\in[r]}\right] \\
&\leq m^t \cdot m^r \cdot 2^{-s} \ \leq\ \frac{1}{4}.
\end{aligned}$$

Overall, the probability that $(L_x)_{x \in f^{-1}(*)}$ is not scattered is less than $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$. □

Note that the condition of being scattered can be checked in polynomial time. Indeed, for each $x \in f^{-1}(*)$, one can enumerate all the elements of $L_x$, which are at most polynomially many in the input size $m^{O(t)}$. Thus, our zero-error randomized reduction picks random linear subspaces $(L_x)_{x \in f^{-1}(*)}$ until we obtain a scattered collection of linear subspaces.

In the rest of the proof, we can thus assume that $(L_x)_{x \in f^{-1}(*)}$ is scattered. The next claim gives the reverse inequality of Claim 9.19.

**Claim 9.22.** $\mathsf{DNF}_{\mathsf{MOD}_m}(g) \geq \mathsf{DNF}_{\mathsf{MOD}_m}(f) + |f^{-1}(*)|$ *if* $(L_x)_{x \in f^{-1}(*)}$ *is scattered.*

Let $C = \bigvee_{k=1}^{K} C_k$ be a minimum $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit computing $g$. (In particular, $K = \mathsf{DNF}_{\mathsf{MOD}_m}(g) \leq \mathsf{DNF}_{\mathsf{MOD}_m}(f) + |f^{-1}(*)| \leq m^{t+1}$.) For each $x \in f^{-1}(*)$, we first extract a subcircuit $C_{l(x)}$ that covers (a large fraction of) the random linear subspace $L_x$. Let $l(x) \in [K]$ be one of the indices such that $|C_{l(x)}^{-1}(1) \cap (\{x\} \times L_x)|$ is maximized. That is, $C_{l(x)}$ covers the largest fraction of the affine subspace $\{x\} \times L_x$; in particular, since $\bigcup_{k\in[K]} C_k^{-1}(1) \supseteq \{x\} \times L_x$, there are at least $|L_x|/K$ $\left(= m^r/K \geq m^{r-t-1} \geq 2\right)$ points in the set $C_{l(x)}^{-1}(1) \cap (\{x\} \times L_x)$. Intuitively, the subcircuits $\{\, C_{l(x)} \mid x \in f^{-1}(*) \,\}$ are supposed to cover random linear subspaces, and the rest of the subcircuits computes $f$.

To make the intuition formal, we will prove the following two claims. The first asserts that, under our constraints, no affine subspace can cover a large fraction of two distinct random linear subspaces.

**Claim 9.23.** $l \colon f^{-1}(*) \to [K]$ *is injective.*

[3]Note that any equation $ax = b \pmod m$ with $a \neq 0$ is satisfied with probability $\leq 1/2$ over a random choice of $x$.

The second claim asserts that, if an affine subspace $C_{l(x')}^{-1}(1)$ covers a large fraction of $\{x'\} \times L_{x'}$, then it cannot cover a point $(x, 0^s)$ such that $f(x) = 1$.

**Claim 9.24.** *$C_{l(x')}(x, 0^s) = 0$ for any $x \in f^{-1}(1)$ and $x' \in f^{-1}(*)$.*

Assuming these two claims, it is easy to prove Claim 9.22.

*Proof of Claim 9.22.* For each $k \in [K]$, define an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit $C'_k$ as $C'_k(x) := C_k(x, 0^s)$ on input $x \in \mathbb{Z}_m^t$. Define a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit $C' := \bigvee_{k \in [K] \setminus \{ l(x) | f(x) = * \}} C'_k$. By Claim 9.23, the number of subcircuits in $C'$ is $K - |f^{-1}(*)|$.

We claim that $C'$ computes $f$. Indeed, for any $x \in f^{-1}(1)$, we have $C(x, 0^s) = g(x, 0^s) = f(x) = 1$; hence, there is some $k \in [K]$ such that $C_k(x, 0^s) = 1$, which implies that $C'_k(x) = 1$ by the definition of $C'_k$. Claim 9.24 implies $k \notin \{ l(x') \mid f(x') = * \}$; thus $C'(x) = 1$. On the other hand, for any $x \in f^{-1}(0)$, we have $C(x, 0^s) = g(x, 0^s) = f(x) = 0$; in particular, for any $k \in [K]$, $C_k(x, 0^s) = 0$. Thus $C'_k(x) = 0$ for any $k \in [K]$, which implies $C'(x) = 0$. □

It remains to prove Claim 9.23 and 9.24. We prove the latter fist.

*Proof of Claim 9.24.* Assume, by way of a contradiction, that $C_{l(x')}(x, 0^s) = 1$ for some $x \in f^{-1}(1)$ and $x' \in f^{-1}(*)$. By the definition of $l(x')$, there are at least 2 distinct points $(x', a)$ and $(x', b)$ in $C_{l(x')}^{-1}(1) \cap (\{x'\} \times L_{x'})$. Since $C_{l(x')}^{-1}(1)$ is an affine subspace, we have $(x', a) - (x', b) + (x, 0^s) = (x, a - b) \in C_{l(x')}^{-1}(1)$ (as in the proof of Claim 9.13). It follows that $C(x, a - b) = 1$. Since $C$ computes $g$, we also have $g(x, a - b) = 1$, which contradicts the fact that $a - b \neq 0^s$ and the definition of $g$. □

*Proof of Claim 9.23.* Assume that $l(x_1) = l(x_2) =: k$ for distinct inputs $x_1, x_2 \in f^{-1}(*)$. Take any 2 distinct points $(x_1, a)$ and $(x_1, b)$ from $C_k^{-1}(1) \cap (\{x_1\} \times L_{x_1})$ and any point $(x_2, c)$ from $C_k^{-1}(1) \cap (\{x_2\} \times L_{x_2})$. Since $C_k^{-1}(1)$ is an affine subspace, we have $(x_1, a) - (x_1, b) + (x_2, c) = (x_2, a - b + c) \in C_k^{-1}(1)$. We also have $(x_2, a - b + c) \in \{x_2\} \times L_{x_2}$, since $C_k^{-1}(1) \cap (\{x_2\} \times \mathbb{Z}_m^s) \subseteq g^{-1}(1) \cap (\{x_2\} \times \mathbb{Z}_m^s) = \{x_2\} \times L_{x_2}$. Therefore, $a - b + c \in L_{x_2}$. Since $c \in L_{x_2}$ and this is a linear subspace, it follows that $a - b \in L_{x_2}$. On the other hand, by the definition of $a$ and $b$, we have $0^s \neq a - b \in L_{x_1}$. However, this is a contradiction because $0^s \neq a - b \in L_{x_1} \cap L_{x_2} = \{0^s\}$. □

*Proof of Theorem 9.18.* By Claim 9.19 and 9.22, we obtain $\mathsf{DNF}_{\mathsf{MOD}_m}(g) = \mathsf{DNF}_{\mathsf{MOD}_m}(f) + |f^{-1}(*)|$ for a scattered collection $(L_x)_{x \in f^{-1}(*)}$. Since $s = O(t \log m)$, the truth table of $g$ is of length $m^{t+s} = m^{O(t \log m)}$, which is a polynomial in the input length for every constant $m \geq 2$. Finally, since it is possible to check whether $(L_x)_{x \in f^{-1}(*)}$ is scattered in polynomial time, the reduction is zero-error. □

**On our proof strategy and the restriction to functions over boolean inputs ($m > 2$).** The linear-algebraic and probabilistic techniques employed here naturally suggest to view a set of inputs for the input instance $f$ as a subset of the algebraic structure $\mathbb{Z}_m^n$ (a vector space or module, depending on $m$). In order to establish a similar NP-hardness result with respect to functions on the hypercube and AND-OR-MOD$_m$ circuits, one is tempted to encode elements from the structure $\mathbb{Z}_m^n$ as binary strings, and to consider a bijection $\varphi \colon \mathbb{Z}_m^n \leftrightarrow$

$\Gamma \subseteq \{0,1\}^*$ between vectors and binary strings. However, a binary encoding allows a bottom-layer modular gate to access individual bits of this encoding, and as a consequence, this gate might accept a set $A \subseteq \{0,1\}^*$ that does not correspond under $\varphi$ to the set of solutions of a modular equation over $\mathbb{Z}_m$. When this is the case, our argument no longer works.

Another natural approach would be to restrict the input function to boolean inputs, and to directly view such inputs as elements in $\{0,1\}^n \subseteq \mathbb{Z}_m^n$. Here certain technical difficulties are transferred to our probabilistic analysis involving affine subspaces of $\mathbb{Z}_m^n$, and it is not immediately clear to us how to modify the argument in this case.

For these reasons, when $m > 2$ our techniques do not seem to be directly applicable to functions defined over boolean inputs only, and a more complicated argument might be necessary. Note however that this does not exclude the existence of different and potentially simpler reductions among these and other intermediary problems.

## 9.4 Derandomization and Pseudorandom Generators for $\mathsf{AND} \circ \mathsf{MOD}_m$

In this section, we present a unified way of efficiently derandomizing the zero-error reductions of Section 9.3. The crucial idea is that certain *subconditions* of being nice or scattered can be checked by $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits over $\mathbb{Z}_m^n$; hence, a pseudorandom generator for $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits can be used to derandomize the reductions.

In order to achieve this, we show that there exists a quick pseudorandom generator with logarithmic seed length that fools any $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit (regardless of its size), a result that might be of independent interest.

**Theorem 9.25.** *For every $\epsilon = \epsilon(n) > 0$ and each $m \geq 2$, there exists a quick pseudorandom generator $G = \{G_n \colon [\Gamma_n] \to \mathbb{Z}_m^n\}_{n \in \mathbb{N}}$ that $\epsilon$-fools any $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit over $\mathbb{Z}_m^n$, where $\Gamma_n = \mathsf{poly}(n, 1/\epsilon, m)$ is a positive integer.*

Here we say that, for $\epsilon > 0$ and an integer $m \geq 2$, a function $G_n : [\Gamma_n] \to \mathbb{Z}_m^n$ *$\epsilon$-fools* $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits if $|\mathbb{E}_{\gamma \sim [\Gamma_n]}[C(G_n(\gamma))] - \mathbb{E}_{v \in_R \mathbb{Z}_m^n}[C(v)]| \leq \epsilon$ for every $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit $C$; such a function $G_n$ is called an *$\epsilon$-pseudorandom generator* for $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits. We say that a family $\{G_n\}_{n \in \mathbb{N}}$ of pseudorandom generators is *quick* if $G_n$ can be computed in $\mathsf{poly}(\Gamma_n)$ time. (Recall that $[\Gamma_n]$ denotes the set $\{1, \ldots, \Gamma_n\}$, which means that the seed-length of $G_n$ is logarithmic in $n$, $m$, and $1/\varepsilon$ when its input elements are represented as binary strings.)

### 9.4.1 Derandomizing the Reductions

We defer a proof of Theorem 9.25 to the next subsection, and present its applications first: The pseudorandom generator implies polynomial-time derandomizations of the reductions presented in Section 9.3.

**Theorem 9.26** (Restatement of Theorem 9.1)**.** $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP *is* NP*-hard under polynomial-time many-one reductions.*

Our basic strategy is as follows: Each reduction of Section 9.3 employs random variables that take value on $\mathbb{Z}_m^k$, for different choices of $k$. To derandomize the reductions, we simply replace these random variables by the output of the pseudorandom generator of Theorem 9.25; then we try all possible $\Gamma_n$ seeds of $G_n$, and check whether the generated random variables satisfy the desired condition

(which can be done in polynomial time). Below we give details for each reduction, starting with the second.

**Derandomizing the second reduction.** We start with the reduction from $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$ to $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP. The reduction required a scattered collection of linear subspaces, which is provided by the probabilistic argument of Claim 9.21. Here we present a deterministic construction of such a collection.

**Theorem 9.27.** *For any integer $m \geq 2$, there exists a deterministic algorithm that, on inputs $t$ and $r$, outputs a* scattered *collection of $r$-dimensional linear subspaces $(L_h)_{h\in[H]}$ for $H := m^t$. Specifically,*

1. *$L_h$ is a linear subspace of $\mathbb{Z}_m^s$ for $s := \lceil (2r + 2t) \log m + 2 \rceil$,*
2. *$|L_h| = m^r$, and*
3. *$L_h \cap L_{h'} = \{0^s\}$ for any distinct $h, h' \in [H]$.*

*The running time of the algorithm is $m^{O((r+t)\log m)}$.*

In the proof of Theorem 9.18, we picked random vectors $v_x^1, \ldots, v_x^r \sim \mathbb{Z}_m^s$ and defined $L_x := \mathsf{span}(v_x^1, \ldots, v_x^r)$ for each $x \in f^{-1}(*) \subseteq \mathbb{Z}_m^t$. We take a similar approach, but instead of generating vectors uniformly at random, we use the output of the pseudorandom generator as the source of randomness. Specifically, let $\gamma \in [\Gamma_{rsH}]$ be a seed of the pseudorandom generator of $G_{rsH}$; define vectors $(v_h^1, \ldots, v_h^r)_{h\in[H]} := G_{rsH}(\gamma) \in (\mathbb{Z}_m^{rs})^H$; then, define $L_h := \mathsf{span}(v_h^1, \ldots, v_h^r)$ for each $h \in [H]$. We show that the probabilistic argument of Claim 9.21 still works even if the randomness is replaced in this way:

**Claim 9.28.** *Let $G_{rsH}$ be the pseudorandom generator of Theorem 9.25 with error parameter $\epsilon = 2^{-s}$. Pick a seed $\gamma \sim [\Gamma_{rsH}]$ uniformly at random, and define a collection $(L_h)_{h\in[H]}$ of linear subspaces as above. Then, $(L_h)_{h\in[H]}$ is scattered with nonzero probability.*

*Proof.* Note that union bounds hold for any distribution; hence, by using the union bounds as in Claim 9.21, the probability that $(L_h)_{h\in[H]}$ is not pairwise disjoint is

$$\begin{aligned} &\Pr\left[\ L_h \cap L_{h'} \neq \{0^s\} \text{ for some distinct } h, h' \in [H]\ \right] \\ &\leq \sum_{h\neq h'\in[H]} \sum_{(c_i),(d_i)} \Pr\left[\sum_{i=1}^r c_i v_h^i = \sum_{i=1}^r d_i v_{h'}^i\right], \end{aligned} \tag{9.2}$$

where the second sum is taken over all nonzero coefficient vectors $(c_i)_{i\in[r]}$ and $(d_i)_{i\in[r]}$ with entries $c_i, d_i \in \mathbb{Z}_m$. If the random vectors $(v_h^i)_{h,i}$ were uniformly distributed, the probability in (9.2) could be bounded by $2^{-s}$ as in Claim 9.21; Here the probability is taken over a random seed $\gamma \sim [\Gamma_{rsH}]$ of the pseudorandom generator $G_{rsH}$. The condition that $\sum_{i=1}^r c_i v_h^i = \sum_{i=1}^r d_i v_{h'}^i$ can be checked by some $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit that takes $(v_h^i)_{h,i}$ as input; thus the circuit is $\epsilon$-fooled by the pseudorandom generator; as a consequence, the probability (9.2) is strictly less than $m^{2t} \cdot m^{2r} \cdot (2^{-s} + \epsilon) \leq \frac{1}{2}$.

Similarly,

$$\begin{aligned}
&\Pr[\ |L_h| < m^r \text{ for some } h \in [H]\ ] \\
&\leq \sum_{h\in[H]} \sum_{(c_i)} \cdot \Pr\left[\sum_{i=1}^{r} c_i v_h^i = 0^s\right] \\
&< m^t \cdot m^r \cdot (2^{-s} + \epsilon) \ \leq \ \frac{1}{2}.
\end{aligned}$$

Overall, the probability that $(L_h)_{h\in[H]}$ is not scattered is strictly less than $\frac{1}{2} + \frac{1}{2} = 1$. □

*Proof of Theorem 9.27.* By Claim 9.28, there exists some seed $\gamma \in [\Gamma_{rsH}]$ such that the output $G_{rsH}(\gamma)$ defines a scattered collection $(L_h)_{h\in[H]}$ of linear subspaces. By exhaustively searching all the seeds, one can enumerate all the outputs of $G_{rsH}$ in time $\mathsf{poly}(\Gamma_{rsH}) = \mathsf{poly}(rsH, 2^s, m)$. Moreover, one can check whether $G_{rsH}(\gamma)$ defines a scattered collection for each $\gamma \in [\Gamma_{rsH}]$ in time $\mathsf{poly}(H, m^s)$. Overall, the running time of our construction is $\mathsf{poly}(m^s) = m^{O((r+t)\log m)}$. □

The randomized reduction of Theorem 9.18 can be now derandomized, using the deterministic construction of Theorem 9.27 for $r := t + 2$.

**Corollary 9.29.** *There is a polynomial-time ($m^{O(t\log m)}$ time on input length $O(m^t)$) many-one reduction from* $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$ *to* $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP.

**Derandomizing the first reduction.** We now consider the reduction from the $r$-bounded set cover problem to $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$. Let $[n]$ be the universe, and $\mathcal{S} \subseteq \binom{[n]}{\leq r}$ be an input to the set cover problem. Derandomizing the reduction amounts to a deterministic construction of a *nice* collection $(v^i)_{i\in[n]}$ of vectors. We generate the random vectors using the pseudorandom generator for $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits, and show that the probabilistic argument of Claim 9.16 still works.

**Claim 9.30** (Revised Claim 9.16)**.** *Let $G_{tn}$ be the pseudorandom generator of Theorem 9.25 with error parameter $\epsilon < m^{-t}$. Pick a seed $\gamma \sim [\Gamma_{tn}]$ uniformly at random. Define $(v^1, \ldots, v^n) := G_{tn}(\gamma) \in (\mathbb{Z}_m^t)^n$. If $t \geq r + ((r+2)\log n + \log|\mathcal{S}| + 1)/\log m$, then $(v^i)_{i\in[n]}$ is nice with nonzero probability.*

*Proof.* By using union bounds as in Claim 9.16, it is sufficient to prove

$$n^{r+2} \cdot |\mathcal{S}| \cdot m^r \cdot \Pr\left[v^{j_S} = \sum_{i\in S} d_i v^i - \sum_{i\in I\setminus\{j_S\}} c_i v^i\right] < 1 \tag{9.3}$$

for coefficients $(c_i)_{i\in I}, (d_i)_{i\in S}$ and $j_S \in I \setminus S$, where the probability is taken over a random seed $\gamma$.

The condition $v^{j_S} = \sum_{i\in S} d_i v^i - \sum_{i\in I\setminus\{j_S\}} c_i v^i$ can be checked by an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit that takes $(v^1, \ldots, v^n) \in \mathbb{Z}_m^{tn}$ as input. By Theorem 9.25, we get

$$\Pr\left[v^{j_S} = \sum_{i\in S} d_i v^i - \sum_{i\in I\setminus\{j_S\}} c_i v^i\right] \leq m^{-t} + \epsilon.$$

Consequently, due to our choice of $t$ and using $\epsilon < m^{-t}$, the left-hand side of (9.3) is strictly less than

$$n^{r+2} \cdot |\mathcal{S}| \cdot m^r \cdot 2m^{-t} \leq 1,$$

which completes the proof. □

In particular, there exists some seed $\gamma \in [\Gamma_{tn}]$ such that $(v^1, \dots, v^n) = G_{tn}(\gamma)$ is nice. The number of seeds is at most $\Gamma_{tn} = \mathsf{poly}(tn, 1/\epsilon, m) = \mathsf{poly}(n, m^t) = (nm)^{O(r)}$, which is a polynomial in the input length; hence, in polynomial time, one can try all possible seeds and find a nice collection $(v^i)_{i\in[n]}$ of vectors. Thus the reduction of Theorem 9.10 can be derandomized:

**Corollary 9.31.** $(\mathsf{DNF} \circ \mathsf{MOD}_m)$-MCSP$^*$ *is* $\mathsf{NP}$*-hard under polynomial-time many-one reductions.*

*Proof of Theorem 9.26.* Immediate from Corollary 9.31 and 9.29. □

### 9.4.2 Near-Optimal Pseudorandom Generators for $\mathsf{AND} \circ \mathsf{MOD}_m$

This subsection contains a proof of Theorem 9.25. We assume basic familiarity with concepts from analysis of boolean functions [O'D14]. For simplicity, we first focus on the case of $m = 2$, which admits a simpler proof.

**Proof for $m = 2$.** An *$\epsilon$-biased generator*, introduced by Naor and Naor [NN93], is a pseudorandom generator for $\mathsf{XOR}$ functions. That is, we say that a function $G\colon \{0,1\}^s \to \{0,1\}^n$ is an $\epsilon$-biased generator if $|\mathbb{E}_{x\sim\{0,1\}^n}[\chi_S(x)] - \mathbb{E}_{s\in_R\{0,1\}^s}[\chi_S(G(s))]| \leq \epsilon$ for any $S \subseteq [n]$, where $\chi_S(x) := \bigoplus_{i\in S} x_i$. While this definition only requires the generator to fool $\mathsf{XOR}$ functions, it can be shown that any Boolean function with small $\ell_1$ Fourier norm can be fooled by $\epsilon$-biased generators.

**Lemma 9.32** (see e.g., [DETT09, Lemma 2.5])**.** *Every function $f\colon \{0,1\}^n \to \{0,1\}$ can be $\epsilon\hat{\|}f\hat{\|}_1$ fooled by any $\epsilon$-biased generator. Here, $\hat{\|}f\hat{\|}_1 := \sum_{S\subseteq[n]} |\widehat{f}(S)|$.*

*Proof Sketch.* Use the Fourier expansion $f(x) = \sum_{S\subseteq[n]} \widehat{f}(S)\chi_S(x)$, and apply the triangle inequality. □

Moreover, it is known that any $\mathsf{AND} \circ \mathsf{XOR}$ circuit $f$ has $\hat{\|}f\hat{\|}_1 = 1$.

**Lemma 9.33** (see e.g., [O'D14, Proposition 3.12])**.** $\hat{\|}f\hat{\|}_1 = 1$ *for any Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ computable by a nontrivial $\mathsf{AND} \circ \mathsf{XOR}$ circuit.*

*Proof Sketch.* Let $H + a \subseteq \{0,1\}^n$ be the (nonempty) affine subspace accepted by $f$. Take a basis of $H^\perp$. Write a characteristic function of $f$ using the basis, and expand it to obtain a Fourier expansion of $f$. □

Combining these two lemmas, any $\epsilon$-biased generator fools $\mathsf{AND}\circ\mathsf{XOR}$ circuits. Moreover, Naor and Naor [NN93] gave an explicit construction of an $\epsilon$-biased generator of seed length $O(\log n + \log(1/\epsilon))$, from which Theorem 9.25 follows when $m = 2$.

In the proof sketched above, we exploited the fact that $\{0,1\}^n = \mathbb{Z}_2^n$ is a vector space: We took a basis of a linear subspace in the proof of Lemma 9.33. In order to generalize the result to the case of $m \geq 2$, we need a more direct proof which does not rely on a basis.

**Proof for any $m \geq 2$.** Azar, Motwani and Naor [AMN98] generalized the notion of $\epsilon$-biased generator on $\{0,1\}^n$ to $\mathbb{Z}_m^n$ for any integer $m \geq 2$, and gave an explicit construction. We review the generalized notion and their result below.

**Definition 9.34** ([AMN98])**.** *For a probability distribution $\mathcal{D}$ over $\mathbb{Z}_m^n$ and a vector $a \in \mathbb{Z}_m^n$, $\mathsf{bias}_{\mathcal{D}}(a)$ is defined as follows: for $g := \gcd(a_1, \ldots, a_n, m)$,*

$$\mathsf{bias}_{\mathcal{D}}(a) := \frac{1}{g} \max_{0 \leq k < m/g} \left| \Pr_{x \sim \mathcal{D}} [\langle a, x \rangle = kg] - \frac{g}{m} \right|.$$

*We say that a distribution $\mathcal{D}$ is* $\epsilon$-biased *if $\mathsf{bias}_{\mathcal{D}}(a) \leq \epsilon$ for every $a \in \mathbb{Z}_m^n$. We say that a function $G \colon [\Gamma] \to \mathbb{Z}_m^n$ is an* $\epsilon$-biased generator *if the distribution $G(\gamma)$ for a random seed $\gamma \sim [\Gamma]$ is $\epsilon$-biased.*

**Theorem 9.35** ([AMN98, Theorem 6.1])**.** *For $m(n) \geq 2$ and $\epsilon = \epsilon(n) > 0$, there exists a quick $\epsilon$-biased generator $G = \{G_n : [\Gamma_n] \to \mathbb{Z}_m^n\}_{n \in \mathbb{N}}$ for some $\Gamma_n = \mathsf{poly}(n, 1/\epsilon, m)$.*

We use the same pseudorandom generator $G$ as in Theorem 9.35. In what follows, we will show that any $\epsilon$-biased generator $m\epsilon$-fools $\mathsf{AND} \circ \mathsf{MOD}_m$ circuits, which completes the proof of Theorem 9.25.

Define $e_m \colon \mathbb{Z}_m \to \mathbb{C}^\times$ as $e_m(k) := \exp(2\pi\sqrt{-1} \cdot k/m)$ for $k \in \mathbb{Z}_m$.

**Lemma 9.36.** *For any distribution $\mathcal{D}$ on $\mathbb{Z}_m^n$ and any nonzero vector $a \in \mathbb{Z}_m^n$, we have*

$$\left| \mathop{\mathbb{E}}_{x \sim \mathcal{D}} [e_m(\langle a, x \rangle)] \right| \leq m \cdot \mathsf{bias}_{\mathcal{D}}(a).$$

*Proof.* The proof follows the same approach of [AMN98, Lemma 4.4]. Let $g := \gcd(a_1, \ldots, a_n, m)$.

$$\begin{aligned}
\left| \mathop{\mathbb{E}}_{x \sim \mathcal{D}} [e_m(\langle a, x \rangle)] \right| &= \left| \sum_{0 \leq k < m/g} e_m(kg) \Pr_{x \sim \mathcal{D}} [\langle a, x \rangle = kg] \right| \\
&= \left| \sum_{0 \leq k < m/g} e_m(kg) \left( \Pr_{x \sim \mathcal{D}} [\langle a, x \rangle = kg] - \frac{g}{m} \right) \right| \\
&\leq \sum_{0 \leq k < m/g} |e_m(kg)| \cdot \left| \Pr_{x \sim \mathcal{D}} [\langle a, x \rangle = kg] - \frac{g}{m} \right| \\
&\leq \frac{m}{g} \cdot 1 \cdot g \cdot \mathsf{bias}_{\mathcal{D}}(a) \;=\; m \cdot \mathsf{bias}_{\mathcal{D}}(a),
\end{aligned}$$

where the first equality follows from the fact that $\langle a, x \rangle$ is a multiple of $g$ for any $x \in \mathbb{Z}_m^n$, and in the second equality we used that $\sum_{0 \leq k < m/g} e_m(kg) = 0$ for $g < m$, which is true if $a \neq 0^n$. □

As a consequence of the previous lemma, we can prove that any affine function can be "fooled":

**Lemma 9.37.** *For any $\epsilon$-biased probability distribution $\mathcal{D}$ on $\mathbb{Z}_m^n$, any vector $a \in \mathbb{Z}_m^n$, and any scalar $b \in \mathbb{Z}_m$,*

$$\left| \mathop{\mathbb{E}}_{x \sim \mathcal{D}} [e_m(\langle a, x \rangle + b)] - \mathop{\mathbb{E}}_{x \sim \mathbb{Z}_m^n} [e_m(\langle a, x \rangle + b)] \right| \leq m\epsilon.$$

*Proof.* When $a = 0^n$, both expectations are constant, and hence the lemma follows. Otherwise, we have $\mathbb{E}_{x\sim\mathbb{Z}_m^n}[e_m(\langle a, x\rangle)] = 0$, since this expression can be written as a product of expectations, and one of them evaluates to zero. Using Lemma 9.36, we obtain

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[e_m(\langle a, x\rangle + b)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Z}_m^n}[e_m(\langle a, x\rangle + b)]\right| \\
&= |e_m(b)| \cdot \left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[e_m(\langle a, x\rangle)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Z}_m^n}[e_m(\langle a, x\rangle)]\right| \;=\; 1 \cdot \left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[e_m(\langle a, x\rangle)]\right| \\
&\leq\; m\,\mathsf{bias}_{\mathcal{D}}(a) \;\leq\; m\epsilon.
\end{aligned}$$

□

**Theorem 9.38.** *For any $\epsilon$-biased probability distribution $\mathcal{D}$ on $\mathbb{Z}_m^n$ and any function $f : \mathbb{Z}_m^n \to \{0, 1\}$ computable by some $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit,*

$$\left|\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Z}_m^n}[f(x)]\right| \leq m\epsilon$$

*Proof.* Suppose that an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit computing $f$ has $K$ $\mathsf{MOD}_m$ gates, and, for each $k \in [K]$, let $g_k \colon \mathbb{Z}_m^n \to \mathbb{Z}_m$ denote the *affine function* that corresponds to the $k$th $\mathsf{MOD}_m$ gate. That is, $g_k(x) = \langle a_k, x\rangle + b_k$ for some vector $a_k \in \mathbb{Z}_m^n$ and some scalar $b_k \in \mathbb{Z}_m$; moreover, for any input $x \in \mathbb{Z}_m^n$, $f(x) = 1$ if and only if $g_k(x) = 0$ for all $k \in [K]$.

We employ the following construction. Let $p(z)$ be the polynomial over $\mathbb{C}$ defined as follows.

$$p(z) := \frac{1}{m} \prod_{\alpha\in\mathbb{Z}_m\setminus\{0\}} (z - e_m(\alpha)) \tag{9.4}$$

$$= \frac{1}{m}\frac{z^m - 1}{z - 1} \;=\; \frac{1}{m}\sum_{i=0}^{m-1} z^i, \tag{9.5}$$

where the second equality holds because the roots of the polynomial $z^m - 1$ are $\{\, e_m(\alpha) \mid \alpha \in \mathbb{Z}_m \,\}$. Useful properties of this polynomial are that, by (9.4), we have $p(e_m(\alpha)) = 0$ for any $\alpha \in \mathbb{Z}_m \setminus \{0\}$, and that $p(e_m(0)) = p(1) = 1$ because of (9.5). Using the polynomial, we can write $f$ as follows:

$$\begin{aligned}
f(x) &= \bigwedge_{k\in[K]} [g_k(x) = 0] \\
&= \bigwedge_{k\in[K]} [p(e_m(g_k(x))) = 1] \\
&= \prod_{k\in[K]} p(e_m(g_k(x))) \\
&= \prod_{k\in[K]} \left(\frac{1}{m}\sum_{j=0}^{m-1} e_m(j \cdot g_k(x))\right) \\
&= \frac{1}{m^K} \prod_{k\in[K]} \sum_{\alpha_k\in\mathbb{Z}_m} e_m(\alpha_k g_k(x)) \\
&= \frac{1}{m^K} \sum_{\alpha\in\mathbb{Z}_m^K} e_m\left(\sum_{k\in[K]} \alpha_k g_k(x)\right).
\end{aligned}$$

Now, by using Lemma 9.37, we obtain

$$\begin{aligned}
&\left| \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[f(x)] - \mathop{\mathbb{E}}_{x\sim\mathbb{Z}_m^n}[f(x)] \right| \\
&\leq \frac{1}{m^K} \sum_{\alpha\in\mathbb{Z}_m^K} \left| \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \left[ e_m \left( \sum_{k\in[K]} \alpha_k g_k(x) \right) \right] - \mathop{\mathbb{E}}_{x\sim\mathbb{Z}_m^n} \left[ e_m \left( \sum_{k\in[K]} \alpha_k g_k(x) \right) \right] \right| \\
&\leq m\epsilon,
\end{aligned}$$

where in the last inequality we used the fact that $\sum_{k\in[K]} \alpha_k g_k(x)$ is an affine function. □

*Proof of Theorem 9.25.* The result is immediate from Theorem 9.35 and 9.38. □

## 9.5 Appendix

### 9.5.1 Proof of Fact 9.5 – Double Orthogonal Complement in $(\mathbb{Z}/m\mathbb{Z})^n$

In this section we present the proof of Fact 9.5, which for convenience is reformulated as Theorem 9.39 stated below. Our presentation follows the proof outlined in [God].

Recall the following concepts. We consider the Abelian group $G := (\mathbb{Z}/m\mathbb{Z})^n$ equipped with component-wise addition modulo $m$, and let $\langle x, y\rangle := \sum_{i\in[n]} x_i y_i \bmod m$, where $x, y \in G$. For a subgroup $V$ of $G$, define $V^\perp := \{\, x \in G \mid \langle x, y\rangle = 0 \text{ for all } y \in V \,\}$, which is again a subgroup of $G$.

**Theorem 9.39** (folklore)**.** *$V^{\perp\perp} = V$ for any subgroup $V$ of $G = (\mathbb{Z}/m\mathbb{Z})^n$.*

It is easy to see $V \subseteq V^{\perp\perp}$: indeed, for any $x \in V$, we have $\langle x, y\rangle = 0$ for each $y \in V^\perp$ by the definition of $V^\perp$; hence $x \in V^{\perp\perp}$. Therefore, it is sufficient to show that the size of $V^{\perp\perp}$ is equal to that of $V$. To this end, we prove the following claim.

**Claim 9.40.** *$|V^\perp| = |G|/|V|$ for any subgroup $V$ of $G$.*

Note that, applying this claim twice, we obtain $|V^{\perp\perp}| = |G|/|V^\perp| = |G|/(|G|/|V|) = |V|$, which completes the proof of Theorem 9.39. Claim 9.40 will be proved by combining the three claims below.

Let $H$ be any finite Abelian group. A *character* of the group $H$ is a homomorphism $\chi\colon H \to \mathbb{C}^\times$ Let $\widehat{H}$ denote the dual group of $H$, that is, the group of all characters of $H$. (See e.g. [O'D14, Section 8.5] for more details.) It is known that the order of a group $H$ and the order of its dual group $\widehat{H}$ are the same.

**Claim 9.41** ([O'D14, Corollary of Proposition 8.55 and Exercise 8.35])**.** *$|H| = |\widehat{H}|$ for any finite Abelian group $H$.*

For any subgroup $V$ of $G$, define $V^* := \{\, \chi \in \widehat{G} \mid \chi(v) = 1 \text{ for every } v \in V \,\}$.

**Claim 9.42.** *$\widehat{G/V} \cong V^*$ for any subgroup $V$ of $G$.*

*Proof.* We define an isomorphism $\varphi\colon \widehat{G/V} \to V^*$. Given $\chi \in \widehat{G/V}$, we define $\varphi(\chi)\colon G \to \mathbb{C}^\times$ by $\varphi(\chi)(x) := \chi(x + V)$ for $x \in G$. We claim that $\varphi(\chi)$ is indeed in $V^*$: First, $\varphi(\chi)\colon G \to \mathbb{C}^\times$ is a homomorphism since $\varphi(\chi)(x + y) = \chi(x + y + V) = \chi((x + V) + (y + V)) = \chi(x + V)\chi(y + V)$ for any $x, y \in G$.

Second, $\varphi(\chi)(v) = \chi(v+V) = \chi(V) = 1$ for any $v \in V$. (Here, we used the fact that the homomorphism $\chi$ maps the identity $0+V \in G/V$ to the identity $1 \in \mathbb{C}^\times$.)

We claim that $\varphi$ is a homomorphism. Indeed, $\varphi(\chi_1\chi_2)(x) = (\chi_1\chi_2)(x+V) = \chi_1(x+V)\chi_2(x+V) = \varphi(\chi_1)(x)\varphi(\chi_2)(x)$ for any $x \in G$ and any $\chi_1, \chi_2 \in \widehat{G/V}$; hence $\varphi(\chi_1\chi_2) = \varphi(\chi_1)\varphi(\chi_2)$.

In order to prove that $\varphi$ is a bijection, we construct an inverse map $\psi\colon V^* \to \widehat{G/V}$. Given $\chi \in V^*$, define $\psi(\chi)(a+V) := \chi(a)$ for any coset $a+V \in G/V$. Note that this map is well defined since $a+V = b+V$ implies $a-b \in V$, and thus $1 = \chi(a-b) = \chi(a)/\chi(b)$. It is straightforward to see that $\psi = \varphi^{-1}$: indeed, $\psi(\varphi(\chi))(a+V) = \varphi(\chi)(a) = \chi(a+V)$ and $\varphi(\psi(\chi))(a) = \psi(\chi)(a+V) = \chi(a)$ for any $a \in G$. Hence $\varphi$ is both injective and surjective, and consequently, an isomorphism. □

**Claim 9.43.** *$V^* \cong V^\perp$ for any subgroup $V$ of $G = (\mathbb{Z}/m\mathbb{Z})^n$.*

*Proof.* We first prepare some notation: For any $i \in [n]$, let $e_i \in G$ be the vector whose value is 1 on the $i$th coordinate and is 0 on the other coordinates. Let $\omega := \exp(2\pi\sqrt{-1}/m) \in \mathbb{C}^\times$ denote the $m$th root of unity.

We construct an isomorphism $\varphi\colon V^\perp \to V^*$. Given $x \in V^\perp$, define $\varphi(x) \in V^*$ as $\varphi(x)(y) := \omega^{\langle x,y\rangle}$ for any $y \in G$. Note that the image of $\varphi$ is contained in $V^*$: indeed, for any $v \in V^\perp$, we have $\varphi(x)(v) = \omega^{\langle x,v\rangle} = \omega^0 = 1$.

We claim that $\varphi$ is injective. It is easy to see that $\varphi$ is a homomorphism; thus, it is sufficient to prove that the kernel of $\varphi$ is just $0 \in V^\perp$. If $\varphi(x)$ is the constant function 1, then $\langle x, y\rangle = 0$ for any $y \in G$; in particular, letting $y \in \{e_1, \dots, e_n\}$, we obtain $x = 0$.

Finally, we claim that $\varphi$ is surjective. For any $\chi \in V^*$ and any $i \in [n]$, there is some $x_i \in \mathbb{Z}/m\mathbb{Z}$ such that $\chi(e_i) = \omega^{x_i}$: indeed, since $1 = \chi(0) = \chi(m \cdot e_i) = \chi(e_i)^m$, $\chi(e_i)$ is one of the $m$th roots of unity. Now we define $x := \sum_{i=1}^n x_i e_i \in G$. Then, for any $y \in G$, $\varphi(x)(y) = \omega^{\langle x,y\rangle} = \prod_{i=1}^n \omega^{x_i y_i} = \prod_{i=1}^n \chi(e_i)^{y_i} = \prod_{i=1}^n \chi(y_i e_i) = \chi(\sum_{i=1}^n y_i e_i) = \chi(y)$; hence $\varphi(x) = \chi$ for some $x \in G$. Moreover, for any $v \in V$, we have $\chi(v) = \omega^{\langle x,v\rangle} = 1$ since $\chi \in V^*$; thus we have $\langle x, v\rangle = 0$, which implies that $x \in V^\perp$. □

Combining these three claims, we obtain $|V^\perp| = |V^*| = |\widehat{G/V}| = |G/V| = |G|/|V|$, which completes the proof of Claim 9.40.

### 9.5.2 On Different Complexity Measures for $\mathsf{DNF} \circ \mathsf{MOD}_p$ Circuits

In this section, we provide an example of the robustness of our arguments with respect to variations of the complexity measure. Let $p \geq 2$ be a fixed prime. We sketch the proof of a hardness result for a variant of the $(\mathsf{DNF} \circ \mathsf{MOD}_p)$-MCSP$^*$ problem, described as follows. We consider layered $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{MOD}_p$ formulas[4] over $\mathbb{Z}_p^n$, and measure complexity by the total number of (non-input) gates in the formula.[5] A bit more precisely, we adapt the proof of Theorem 9.10 from Subsection 9.3.1, and show that this problem is also NP-hard under randomized reductions.

[4]Recall that in a formula every non-input gate has fan-out one.

[5]Under our notion of layered formulas, an $(\mathsf{AND} \circ \mathsf{MOD}_p)$-circuit with a single $\mathsf{MOD}_p$ gate has size 2. While this is convenient for the exposition, it is not particularly important for the result.

Since $\mathbb{Z}_p^t$ is a vector space over the field $\mathbb{Z}_p$, we can define the dimension of an affine subspace: For a linear subspace $H \subseteq \mathbb{Z}_p^t$, let $\dim(H)$ denote the dimension of $H$, and let $\operatorname{codim}(H) := \dim(H^\perp) = t - \dim(H)$; then, for any $a \in \mathbb{Z}_p^t$, define the dimension of an affine subspace $H + a$ as $\dim(H + a) := \dim(H)$, and $\operatorname{codim}(H + a) := \dim(H)$. Observe that this notion is well-defined. Using dimension, we can characterize the number of gates in $\mathsf{AND} \circ \mathsf{MOD}_p$ formulas.

**Lemma 9.44.** *Let $A$ be an affine subspace of $\mathbb{Z}_p^t$. Then, the minimum number of gates in any layered $\mathsf{AND} \circ \mathsf{MOD}_p$ formula accepting $A$ is exactly $1 + \operatorname{codim}(A)$.*

*Proof Sketch.* As in the proof of Lemma 9.4, a layered $\mathsf{AND} \circ \mathsf{MOD}_p$ formula $C$ with $1 + s$ gates accepts the set $A = C^{-1}(1)$ of solutions of $s$ linear equations over $\mathsf{MOD}_p$. Let $B \in \mathbb{Z}_p^{s \times t}$ be the matrix that defines these linear equations. Then, we have $\dim \ker(B) = \dim(A)$, and by the rank-nullity theorem, we obtain $\operatorname{codim}(A) = t - \dim(A) = t - \dim \ker(B) = \operatorname{rank}(B) \leq s$.

Conversely, let $A =: H + a$ for some linear subspace $H$ and some $a \in \mathbb{Z}_p^t$, and let $\gamma_1, \ldots, \gamma_s$ be a basis of $H^\perp$, where $s := \operatorname{codim}(H)$. Then, using orthogonal complements, it is easy to check that $x \in A$ if and only if $\langle \gamma_i, x \rangle = \langle \gamma_i, a \rangle$ for all $i \in [s]$. The latter condition can be written as an $\mathsf{AND} \circ \mathsf{MOD}_p$ layered formula with $1 + s$ gates. □

As a corollary, for any *optimal* layered $(\mathsf{DNF} \circ \mathsf{MOD}_p)$-formula $C = \bigvee_{k=1}^K C_k$ for a function $f \colon \mathbb{Z}_p^n \to \{0, 1\}$, where $C_k$ is an $\mathsf{AND} \circ \mathsf{MOD}_p$ circuit for each $k \in [K]$, the total number of gates in the formula is precisely $1 + K + \sum_{k=1}^K \operatorname{codim}(C_k^{-1}(1))$.

For convenience, given a function $f \colon \mathbb{Z}_p^t \to \{0, 1, *\}$, let $\mathsf{size}(f)$ denote the complexity of $f$ according to our size measure. Now let us revise the proof of Theorem 9.10. Given an instance $\mathcal{S} \subseteq \binom{n}{\leq r}$ of the $r$-bounded set cover instance, we construct a function $f \colon \mathbb{Z}_p^t \to \{0, 1, *\}$ in exactly the same way. Below we adapt the corresponding claims from Subsection 9.3.1. Then we employ the new claims to argue that the NP-hardness result still holds.

**Claim 9.45** (Adaptation of Claim 9.11)**.** *Assume that $\mathcal{S}$ has a set cover of size $K$. Then $\mathsf{size}(f) \leq (t + 1)K + 1$.*

*Proof.* Let $\mathcal{C} \subseteq \mathcal{S}$ be a set cover of size $K$. For each $S \in \mathcal{C}$, let $C_S$ be an $\mathsf{AND} \circ \mathsf{MOD}_p$ circuit over $\mathbb{Z}_p^t$ that accepts $\mathsf{span}(v^S)$. Define a $\mathsf{DNF} \circ \mathsf{MOD}_p$ circuit $C := \bigvee_{S \in \mathcal{C}} C_S$. Then the circuit size of $C$ is $1 + K + \sum_{i=1}^K \operatorname{codim}(C_S^{-1}(1))$, which is obviously at most $1 + K(t + 1)$. □

**Claim 9.46** (Adaptation of Claim 9.15)**.** *Let $(v^i)_{i \in [n]}$ be nice, and $s := \mathsf{size}(f)$. Then $\mathcal{S}$ has a set cover of size $2(s - 1)/(t - r - (\log |\mathcal{S}| / \log p) + 1)$.*

*Proof.* Let $C = \bigvee_{k=1}^K C_k$ be an optimal $\mathsf{DNF} \circ \mathsf{MOD}_p$ layered formula of size $s$ computing $f$. Then, as discussed above, we have $s = 1 + K + \sum_{k=1}^K \operatorname{codim}(C_k^{-1}(1))$. On the other hand, the same analysis from Claim 9.15 shows that $\mathcal{S}$ has a set cover of size $\leq 2K$. It thus remains to give an upper bound on $K$.

Since $C$ computes $f$, we have $C_k^{-1}(1) \subseteq C^{-1}(1) \subseteq f^{-1}(\{1, *\}) = \bigcup_{S \in \mathcal{S}} \mathsf{span}(v^S)$. By counting the number of elements in $C_k^{-1}(1)$ and $\bigcup_{S \in \mathcal{S}} \mathsf{span}(v^S)$, we obtain $p^{\dim(C_k^{-1}(1))} \leq |\mathcal{S}| \cdot p^r$. Hence, we have $\operatorname{codim}(C_k^{-1}(1)) \geq t - r - \log |\mathcal{S}| / \log p$; therefore,

$$s \;\geq\; 1 + K + \sum_{k=1}^{K} \operatorname{codim}(C_k^{-1}(1)) \;\geq\; 1 + K + K(t - r - \log |\mathcal{S}| / \log p),$$

which implies $K \le (s-1)/(t-r-(\log|\mathcal{S}|/\log p)+1)$. □

Let $K$ be the minimum size of a cover for $\mathcal{S}$. By the claims above, we have $\mathsf{size}(f) \lesssim tK$ and $K \lesssim 2\mathsf{size}(f)/t$, because $t$ can be taken large enough compared to the other relevant parameters; hence $\mathsf{size}(f)/t$ roughly gives us a 2-factor approximation. More precisely, we have $\mathsf{size}(f) \le (t+1)K+1 \le 2(t+1)K$, and $K \le 2(\mathsf{size}(f)-1)/((t+1)/2) \le 4\mathsf{size}(f)/(t+1)$ for any $t \ge 2r+2\log|\mathcal{S}|/\log p-1$. That is, the set cover size $K$ satisfies

$$\frac{\mathsf{size}(f)}{2(t+1)} \;\le\; K \;\le\; \frac{4\mathsf{size}(f)}{t+1},$$

which gives an 8-factor approximation of $K$. Since we can take $r$ to be a sufficiently large constant in Theorem 9.7, the result holds.

### 9.5.3 A Hardness of Approximation Result for $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP

The reduction from $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP$^*$ to $(\mathsf{DNF}\circ\mathsf{MOD}_m)$-MCSP presented in Section 9.3 is not *approximation-preserving*: given a partial function $f\colon \mathbb{Z}_m^t \to \{0,1,*\}$, it produces a total function $g\colon \mathbb{Z}_m^{O(t\log m)} \to \{0,1\}$ such that $\mathsf{DNF}_{\mathsf{MOD}_m}(g) = \mathsf{DNF}_{\mathsf{MOD}_m}(f) + |f^{-1}(*)|$. The reduction introduces an additive term $|f^{-1}(*)|$, and hence a (multiplicative) approximation of $\mathsf{DNF}_{\mathsf{MOD}_m}(g)$ does not give a good approximation of $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$. In order to fix this situation, we give an approximation-preserving reduction. Our approach is inspired by a reduction described in [AHM+08].

**Theorem 9.47** (Approximation-preserving version of Corollary 9.29)**.** *There is a polynomial-time algorithm that, given the truth table of a partial function $f\colon \mathbb{Z}_m^t \to \{0,1,*\}$, produces the truth table of a total function $g\colon \mathbb{Z}_m^{2t+2s} \to \{0,1\}$ such that*

$$\mathsf{DNF}_{\mathsf{MOD}_m}(g) = |f^{-1}(*)| \cdot (\mathsf{DNF}_{\mathsf{MOD}_m}(f)+1),$$

*where $s := \lceil (6t+4)\log m + 2 \rceil$.*

*Proof.* The idea of the proof is to amplify the circuit size for $f$; that is, we would like to force any circuit $C$ computing $g$ to also compute sub-functions corresponding to $|f^{-1}(*)|$ copies of $f$.

We can amplify the circuit size as follows. Let $(L_x)_{x\in f^{-1}(*)}$ be a scattered collection of linear subspaces of $\mathbb{Z}_m^s$. Define a function $g'$ by $g'(x,z,w) := f(x)$ if $z \in f^{-1}(*)$ and $w \in L_z$; otherwise $g'(x,z,w) := 0$. Then, under an appropriate choice of parameters, it can be shown that $\mathsf{DNF}_{\mathsf{MOD}_m}(g') = |f^{-1}(*)| \cdot \mathsf{DNF}_{\mathsf{MOD}_m}(f)$. By combining an analogous reduction and the idea behind the proof of Theorem 9.18, we can obtain a total function $g$ such that $\mathsf{DNF}_{\mathsf{MOD}_m}(g) = \mathsf{DNF}_{\mathsf{MOD}_m}(g') + |f^{-1}(*)| = |f^{-1}(*)| \cdot (\mathsf{DNF}_{\mathsf{MOD}_m}(f)+1)$.[6] Details follow.

We first obtain a scattered collection $(L_x)_{x\in f^{-1}(*)}$ of $r$-dimensional linear subspaces of $\mathbb{Z}_m^s$ by using Theorem 9.27 for $r := 2t+2$. Then we define $g\colon \mathbb{Z}_m^{2t+2s} \to \{0,1\}$ as

$$g(x,y,z,w) := \begin{cases} f(x) & (\text{if } f(x)\in\{0,1\} \text{ and } y = 0^s \text{ and } f(z) = * \text{ and } w \in L_z) \\ 1 & (\text{if } f(x) = * \text{ and } y \in L_x) \\ 0 & (\text{otherwise}) \end{cases}$$

[6] A black-box application of Corollary 9.29 produces a function $g$ such that $\mathsf{DNF}_{\mathsf{MOD}_m}(g) = \mathsf{DNF}_{\mathsf{MOD}_m}(g') + |g'^{-1}(*)|$, which is not sufficient for our purpose because $|g'^{-1}(*)|$ is larger than $|f^{-1}(*)|$.

for any $((x,y),(z,w)) \in (\mathbb{Z}_m^s \times \mathbb{Z}_m^t)^2$.

**Claim 9.48** (Analogue of Claim 9.19)**.** $\mathsf{DNF}_{\mathsf{MOD}_m}(g) \leq |f^{-1}(*)| \cdot (\mathsf{DNF}_{\mathsf{MOD}_m}(f) + 1)$.

*Proof.* Suppose that a $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit $C = \bigvee_{k=1}^{K} C_k$ computes $f$. For each $x^* \in f^{-1}(*)$, take an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit $C_{x^*}$ accepting $\{x^*\} \times L_{x^*}$ (by Lemma 9.4). Define

$$C'(x,y,z,w) := \bigvee_{z^* \in f^{-1}(*)} \bigvee_{k=1}^{K} (C_k(x) \wedge y_1 = 0 \wedge \cdots \wedge y_s = 0 \wedge C_{z^*}(z,w)) \vee \bigvee_{x^* \in f^{-1}(*)} C_{x^*}(x,y).$$

It is easy to see that $C'$ computes $g$. □

The rest of the proof is devoted to the reverse direction.

**Claim 9.49** (Analogue of Claim 9.22)**.** $\mathsf{DNF}_{\mathsf{MOD}_m}(g) \geq |f^{-1}(*)| \cdot (\mathsf{DNF}_{\mathsf{MOD}_m}(f) + 1)$.

Let $C = \bigvee_{k=1}^{K} C_k$ be a minimum $\mathsf{DNF} \circ \mathsf{MOD}_m$ circuit computing $g$. In particular, $K = \mathsf{DNF}_{\mathsf{MOD}_m}(g) \leq |f^{-1}(*)| \cdot (\mathsf{DNF}_{\mathsf{MOD}_m}(f) + 1) \leq m^{2t+1}$. For each $x \in f^{-1}(*)$, let $l(x) \in [K]$ be one of the indices such that $|C_{l(x)}^{-1}(1) \cap (\{x\} \times L_x \times \mathbb{Z}_m^{t+s})|$ is maximized. Since $\bigcup_{k \in [K]} C_k^{-1}(1) \supseteq \{x\} \times L_x \times \mathbb{Z}_m^{t+s}$, there are at least $|L_x| \cdot m^{t+s}/K \geq m^{r+t+s}/m^{2t+1} \geq 2$ points in the set $C_{l(x)}^{-1}(1) \cap (\{x\} \times L_x \times \mathbb{Z}_m^{t+s})$.

Define $T_0 := \{\, C_{l(x)} \mid f(x) = * \,\}$. For each $z \in f^{-1}(*)$, let $T_z$ be the set of all $C_k$ such that $k \in [K]$ and $C_k$ accepts at least 2 elements from $\{(x, 0^s, z)\} \times L_z$ for some $x \in f^{-1}(1)$. We will show that the sets $T_0, \{T_z\}_{z \in f^{-1}(*)}$ are pairwise disjoint, and hence $K \geq |T_0| + \sum_{z \in f^{-1}(*)} |T_z|$. We will also prove that $|T_0| = |f^{-1}(*)|$ and $|T_z| \geq \mathsf{DNF}_{\mathsf{MOD}_m}(f)$, which completes the proof.

**Claim 9.50.** *$l \colon f^{-1}(*) \to [K]$ is injective (hence $|T_0| = |f^{-1}(*)|$).*

**Claim 9.51.** *$T_0 \cap T_z = \varnothing$ for any $z \in f^{-1}(*)$.*

Since the proofs of these claims are essentially the same as in Claim 9.23 and 9.24, respectively (except that we have extra coordinates taking values in $\mathbb{Z}_m^t \times \mathbb{Z}_m^s$), we omit them.

**Claim 9.52.** *$T_{z_1} \cap T_{z_2} = \varnothing$ for any distinct elements $z_1, z_2 \in f^{-1}(*)$.*

*Proof.* The proof is basically the argument from Claim 9.23. For completeness, we briefly repeat it here. Towards a contradiction, assume that there exists a circuit $C_k$ in $T_{z_1} \cap T_{z_2}$. By the definition of $T_{z_1}$ and $T_{z_2}$, there exist elements $x_1, x_2 \in f^{-1}(1)$, $a \neq b \in L_{z_1}$, and $c \in L_{z_2}$ such that $C_k(x_1, 0^s, z_1, a) = C_k(x_1, 0^s, z_1, b) = C_k(x_2, 0^s, z_2, c) = 1$. Since $C_k^{-1}(1)$ is an affine subspace, we have $(x_1, 0^s, z_1, a) - (x_1, 0^s, z_1, b) + (x_2, 0^s, z_2, c) = (x_2, 0^s, z_2, a - b + c) \in C_k^{-1}(1)$. Since $C_k^{-1}(1) \cap (\{(x_2, 0^s, z_2)\} \times \mathbb{Z}_m^s) \subseteq \{(x_2, 0^s, z_2)\} \times L_{z_2}$, we get $a - b + c \in L_{z_2}$. However, given that $c \in L_{z_2}$, we obtain $0^s \neq a - b \in L_{z_1} \cap L_{z_2}$, which contradicts $L_{z_1} \cap L_{z_2} = \{0^s\}$. □

Fix any $z \in f^{-1}(*)$. For each $C_k \in T_z$, define an $\mathsf{AND} \circ \mathsf{MOD}_m$ circuit $C'_k$ so that $C_k'^{-1}(1) = \{\, x \in \mathbb{Z}_m^t \mid C_k(x, 0^s, z, w) = 1 \text{ for some } w \in \mathbb{Z}_m^s \,\}$. (Note that a projection of an affine subspace $C_k^{-1}(1)$ is again an affine subspace because a projection is a homomorphism.) Now define $C_z := \bigvee_{C_k \in T_z} C'_k$.

**Claim 9.53.** *$C_z$ computes $f$ for any $z \in f^{-1}(*)$. (In particular, $|T_z| \geq \mathsf{DNF}_{\mathsf{MOD}_m}(f)$.)*

*Proof.* Fix any $x \in f^{-1}(1)$. Since $\{(x, 0^s, z)\} \times L_z$ is covered by $\bigcup_{k\in[K]} C_k^{-1}(1)$, and $|L_z| = m^r$, $K \leq m^{2t+1}$, and $r = 2t + 2$, there exists $k \in [K]$ such that there are at least 2 elements in $(\{(x, 0^s, z)\} \times L_z) \cap C_k^{-1}(1)$; hence, by the definition of $T_z$, we have $C_k \in T_z$. Moreover, $C'_k(x) = 1$ by the definition of $C'_k$; thus $C_z(x) = \bigvee_{C_k\in T_z} C'_k(x) = 1$.

Now fix any $x \in f^{-1}(0)$. Since $g(x, 0^s, z, w) = 0$ for every $w \in \mathbb{Z}_m^s$, we get $C_k(x, 0^s, z, w) = 0$ for any $C_k \in T_z$; thus $C'_k(x) = 0$, which implies that $C_z(x) = 0$. □

Combining the claims above, we obtain

$$\mathsf{DNF}_{\mathsf{MOD}_m}(g) = K \geq |T_0| + \sum_{z\in f^{-1}(*)} |T_z| \geq |f^{-1}(*)| \cdot (\mathsf{DNF}_{\mathsf{MOD}_m}(f) + 1).$$

This completes the proof of Theorem 9.47. □

We can then establish a hardness of approximation result for computing $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$. For a function $f\colon \mathbb{Z}_m^t \to \{0, 1\}$, define $|f| := m^t$, which is the number of entries in the truth table of a function $f$.

**Theorem 9.54.** *There exists a constant $c > 0$ such that if there is a quasipolynomial-time algorithm which approximates $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$ to within a factor of $c \log\log|f|$, then $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{(\log n)^{O(1)}})$.*

*Proof.* As noted by Trevisan [Tre01b], by choosing the parameters of Feige's reduction [Fei98], one can obtain hardness of approximation results for the $r$-bounded set cover problem. While Trevisan only analyzed the case when $r$ is constant (cf. Theorem 9.7), a similar analysis[7] shows that it is $\mathsf{NP}$-hard (under quasipolynomial-time many-one reductions) to approximate the $r(n)$-bounded set cover problem on $n$ points within a factor of $\gamma \log r(n)$ $(= \gamma \log\log n)$ for $r(n) := \log n$ and some small constant $\gamma > 0$.

Suppose that $\mathsf{DNF}_{\mathsf{MOD}_m}(g)$ can be approximated to within a factor of $(\gamma/6) \log\log|g|$ by an algorithm $A$, where $g\colon \mathbb{Z}_m^t \to \{0, 1\}$ is a total function. We show below that if $A$ runs in quasipolynomial time, then $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{(\log n)^{O(1)}})$.

First, note that in order to conclude this it is enough to describe a quasipolynomial-time algorithm $B$ that approximates $r$-Bounded Set Cover to within a factor of $\gamma \log r(n)$ for $r(n) = \log n$. Let $([n], \mathcal{S})$ be an instance of the $r$-Bounded Set Cover Problem. Algorithm $B$ applies the deterministic $n^{O(r(n))}$-time reduction provided by Corollary 9.31 to produce a partial Boolean function $f\colon \mathbb{Z}_m^{O(r\log n)} \to \{0, 1, *\}$. It then invokes the deterministic reduction from Theorem 9.47 to construct from $f$ a total function $g\colon \mathbb{Z}_m^{O(r\log n)} \to \{0, 1\}$. Finally, $B$ uses the approximation algorithm $A$ to compute a $(\gamma/6) \log\log|g|$ approximation to $\mathsf{DNF}_{\mathsf{MOD}_m}(g)$. Let $\widetilde{g} \in \mathbb{N}$ be the value output by $A$. Algorithm $B$ outputs $\widetilde{K} := 2\widetilde{g}/|f^{-1}(*)|$.

[7] Specifically, for the parameters and notation in [Fei98], given a 3CNF-5 formula on $n$ variables, let $k$ be a sufficiently large constant, $m := \sqrt{\log n}$, and $\ell := c \log\log m$ for a large constant $c$. Then the output of Feige's reduction is an instance of the set cover problem on $N$ $(:= m(5n)^\ell)$ points such that each set is of size at most $m2^{O(\ell)} \leq r(N) = \log N$, and the gap between yes instances and no instances is $(1 - \frac{4}{k}) \ln m = \Omega(\log\log N)$.

Note that $B$ runs in quasipolynomial time under our assumptions. It remains to show that it approximates the solution of the original set cover problem within a factor of $\gamma \log\log n$. Let $K$ be the cost of an optimal solution to the initial set cover instance. Recall that $2\mathsf{DNF}_{\mathsf{MOD}_m}(f)$ is a 2-factor approximation for $K$; that is, $K \leq 2 \cdot \mathsf{DNF}_{\mathsf{MOD}_m}(f) \leq 2K$. On the other hand, the guarantees of the algorithm $A$ imply that

$$\mathsf{DNF}_{\mathsf{MOD}_m}(g) \;\leq\; \widetilde{g} \;\leq\; \mathsf{DNF}_{\mathsf{MOD}_m}(g) \cdot (\gamma/6) \log\log |g|.$$

Since $\mathsf{DNF}_{\mathsf{MOD}_m}(g) = |f^{-1}(*)| \cdot (\mathsf{DNF}_{\mathsf{MOD}_m}(f) + 1)$, we get

$$K \;\leq\; \frac{2\widetilde{g}}{|f^{-1}(*)|} \;\leq\; (\gamma/6) \log\log |g| \cdot (K+1)$$

Therefore, for large enough $n$ and on non-trivial instances (i.e. $K \geq 1$), the value $\widetilde{K}$ output by $B$ approximates $K$ to within a factor of $2 \cdot (\gamma/6) \log\log |g| \leq (\gamma/3) \cdot (\log r(n) + \log\log n + O(1)) \leq (\gamma/3) \cdot 3 \log\log n$. □

Finally, we note that when $m$ is prime, it is possible to design a quasipolynomial-time approximation algorithm for $\mathsf{DNF}_{\mathsf{MOD}_m}(f)$ with an approximation factor of $O(\log |f|)$.

**Theorem 9.55.** *Let $p$ be a prime number. There is a quasipolynomial-time algorithm which approximates $\mathsf{DNF}_{\mathsf{MOD}_p}(f)$ to within a factor of $\ln |f|$.*

*Proof.* Let $|f| = p^t$ be the number of entries in the truth table of $f$, the input function. By the results of Subsection 9.2.2, computing $\mathsf{DNF}_{\mathsf{MOD}_p}(f)$ is equivalent to solving a set cover instance. Recall that set cover admits a polynomial-time approximation algorithm that achieves an approximation factor of $\ln N$ on instances over a universe of size $N$ (cf. [Sla97]). Consequently, in order to prove the result it is enough to verify that computing $\mathsf{DNF}_{\mathsf{MOD}_p}(f)$ reduces to a set cover instance with domain size $N_f := |f^{-1}(1)| \leq |f|$ and of size at most quasipolynomial in $|f|$.

Indeed, for a non-zero function $f\colon \mathbb{Z}_p^t \to \{0,1\}$, $\mathsf{DNF}_{\mathsf{MOD}_p}(f)$ is exactly the minimum number of affine subspaces that cover $f^{-1}(1)$. Therefore, by relabelling elements, computing $\mathsf{DNF}_{\mathsf{MOD}_p}(f)$ reduces to a set cover instance $([N_f], \mathcal{S}_f)$, where a set $S \in S_f$ if and only if $S$ viewed as a subset of $\mathbb{Z}_p^t$ is an affine subspace contained in $f^{-1}(1)$. Each such affine subspace has dimension at most $t$, and can be explicitly described by a basis $v_1, \ldots, v_\ell \in \mathbb{Z}_p^t$, where $\ell \leq t$, and a vector $b \in \mathbb{Z}_p^t$. Hence there are at most $p^{O(t^2)}$ such spaces, and consequently, $|\mathcal{S}_f| \leq p^{O(t^2)}$. In other words, we get a set cover instance over a ground set of size $\leq |f|$, and this instance contains at most $|f|^{O(\log |f|)}$ sets.

Finally, since the sets in $\mathcal{S}_f$ can be generated in time at most $|f|^{O(\log |f|)}$, and the set cover approximation algorithm runs in time polynomial in its input length, the result holds. □

# Chapter 10

# Hardness Under Local Reductions

In this chapter, we show that MKTP is hard for the complexity class $\mathsf{DET}$ under *local reductions*, that is, reductions whose output bit depends on a small number of input bits. This is surprising, because prior work on MCSP and MKTP had highlighted weaknesses of "local" reductions such as $\mathsf{DTIME}(n^{0.49})$ reductions. We exploit our local reduction to obtain several new consequences, such as the equivalence between circuit lower bounds and hardness of $\mathrm{MKTP}^A$ for some class of oracles $A$. We start with outlining the contributions of this chapter in the next section.

## 10.1 Background and Overview

### Hardness Under Local Reductions

Murray and Williams [MW17] showed that MCSP and MKTP are not $\mathsf{TC}^0$-hard under "local" reductions, i.e., reductions that are not given enough time to read the whole input. Specifically, they considered a $n^{0.49}$-time reduction such that, given random access to an $n$-bit input and an index $i \in [n^{O(1)}]$, the reduction computes the $i$th bit of the output in $n^{0.49}$ steps, and showed that even $\mathsf{PARITY}$ is not reducible to MCSP under such a local reduction. Murray and Williams speculated that this might be a promising first step toward showing that MCSP is not hard for $\mathsf{NP}$ under Dlogtime-uniform $\mathsf{AC}^0$ reductions, since it follows from [Agr11] that any set that is hard for $\mathsf{TC}^0$ under $\mathsf{P}$-uniform $\mathsf{AC}^0$ reductions is also hard for $\mathsf{TC}^0$ under $\mathsf{P}$-uniform $\mathsf{NC}^0$ reductions. Indeed, the results of Murray and Williams led us to expect that MCSP and MKTP are not even hard for $\mathsf{PARITY}$ under nonuniform $\mathsf{NC}^0$ reductions.

*Contrary to these expectations*, we show that MKTP is hard not only for $\mathsf{TC}^0$ but even for the complexity class $\mathsf{DET}$ under nonuniform $\mathsf{NC}^0$ reductions (Theorem 10.3). Here $\mathsf{DET}$ is the class of problems that are reducible to the problem $\mathsf{Det}$ of computing the determinant of integer matrices, by $\mathsf{NC}^1$-Turing reductions. $\mathsf{DET}$ lies between $\mathsf{L}$ and $\mathsf{P}$. Under a plausible derandomization hypothesis, our nonuniform reduction can be converted into a Dlogtime-uniform $\leq_{\mathrm{tt}}^{\mathsf{AC}^0}$ reduction that is an AND of $\mathsf{NC}^0$-computable queries. Thus "local" reductions are more effective for reductions to MKTP than may have been suspected.

Our $\mathsf{DET}$-hardness result is proved by building on a randomized reduction of [AGvM+18], reducing Graph Isomorphism to MKTP. We modify that construction, to obtain a nonuniform $\mathsf{AC}^0$ reduction (Corollary 10.2). The restricted version of Graph Isomorphism that we use is known to be hard for $\mathsf{DET}$ [Tor04]. Our proof of Theorem 10.3 then appeals to the "Gap Theorem" of [AAR98], in order to conclude that $\mathsf{DET} \leq_{\mathrm{m}}^{\mathsf{NC}^0}$ MKTP; the Gap Theorem states that, for any

class $\mathcal{C}$ closed under $\mathsf{TC}^0$ reductions, $\mathcal{C}$-hardness under $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reductions implies $\mathcal{C}$-hardness under $\leq_{\mathrm{m}}^{\mathsf{NC}^0}$ reductions.

Our hardness results (both unconditional hardness results under nonuniform reductions, and conditional uniform hardness results) are summarized in Table 10.1.

Table 10.1: Hardness for $\mathrm{MKTP}^A$: $\mathrm{MKTP}^A$ is hard for $\mathsf{DET}$ under the type of reducibility listed in the first column, if oracle $A$ satisfies the condition listed in the second column. The last column shows the theorem where the result is stated.

| reductions $\mathcal{R}$ | condition on $A$ | Theorem |
|---|---|---|
| nonuniform $\leq_{\mathrm{m}}^{\mathsf{NC}^0}$ | every $A$ | Theorem 10.3 |
| $\mathsf{P}$-uniform $\leq_{\mathrm{ctt}}^{\mathsf{AC}^0}$ | $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$ | Corollary 10.9 |
| $\mathsf{L}$-uniform $\leq_{\mathrm{ctt}}^{\mathsf{AC}^0}$ | $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$ | Theorem 10.5 |
| Dlogtime-uniform $\leq_{\mathrm{ctt}}^{\mathsf{AC}^0}$ | $\Sigma_d\mathsf{TIME}(n)$ hard on average for $\mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$ | Theorem 10.6 |

### Equivalence Between Circuit Lower Bounds and Hardness

As shown in Chapter 7, hardness of MCSP under deterministic reductions implies circuit lower bounds. It is natural to wonder whether the converse direction holds. That is, does circuit lower bounds imply hardness of MCSP? Note that it is known that if $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ then MCSP is $\mathsf{NP}$-hard under $\mathsf{ZPP}$-Turing reductions [ABK+06b, IKV18]; in this sense, $\mathsf{NP}$-hardness of MCSP is open only in the case when there is a circuit lower bound of $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$.

While we are not able to fully answer the question, we make the partial progress by considering an oracle version of $\mathrm{MKTP}^A$. For any oracle $A$ satisfying $\mathrm{MKTP}^A \subseteq \mathsf{P/poly}$, we show that $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^A(2^{\epsilon n})$ for some $\epsilon > 0$ *if and only if* $\mathrm{MKTP}^A$ is hard for $\mathsf{DET}$ under a certain class of reducibilities. (See Theorem 10.7, and the Remark after Corollary 10.10.)

### Non-Oracle-Independent Reductions

In Chapter 7, we showed that $\mathrm{MCSP}^A \not\leq_{\mathrm{T}}^{\mathsf{P}} \mathrm{MCSP}$ for some oracle $A$ unless $\mathrm{MCSP} \in \mathsf{P}$. The oracle $A$ is constructed by a diagonalization argument, so it is a very artificial. A natural question is whether a similar behavior happens for a natural oracle $A$. Here we show that $\mathsf{DET}$ reduces to MKTP (Theorem 10.5), whereas even $\mathsf{PARITY}$ is not reducible to $\mathrm{MKTP}^{\mathsf{QBF}}$ (Corollary 10.10) under some logspace-uniform $\mathsf{AC}^0$ reductions, assuming a plausible complexity assumption. Thus the power of $\mathrm{MKTP}^A$ oracles is not monotone increasing with respect to even natural oracles $A$ such as $\mathsf{QBF}$ oracles. It should be also noted that the reduction from $\mathsf{DET}$ to MKTP under logspace-uniform $\mathsf{AC}^0$ reductions is not oracle-independent because it does not generalize to a reduction to $\mathrm{MKTP}^A$ for $A = \mathsf{QBF}$. (The part of a proof in which the reduction is not oracle-independent is simply the plausible complexity assumption which we assume in order to derandomize the reduction from $\mathsf{DET}$ to MKTP.)

### Preliminaries: Uniformity of Circuits

In this chapter, we will use various notions of *uniformity* of circuit families. For example, the class $\mathsf{AC}^0$ (corresponding to families $\{C_n : n \in \mathbb{N}\}$ of unbounded

fan-in AND, OR, and NOT gates having size $n^{O(1)}$ and depth $O(1)$) comes in various flavors, depending on the complexity of computing the mapping $1^n \mapsto C_n$. When this is computable in polynomial time (or logarithmic space), then one obtains P-uniform $\mathsf{AC}^0$ (logspace-uniform $\mathsf{AC}^0$, respectively). If no restriction at all is imposed, then one obtains nonuniform $\mathsf{AC}^0$. As discussed in [Vol99], the more restrictive notion of Dlogtime-uniform $\mathsf{AC}^0$ is frequently considered to be the "right" notion of uniformity to use when discussing small complexity classes such as $\mathsf{AC}^0, \mathsf{AC}^0[p]$ and $\mathsf{TC}^0$. If these classes are mentioned with no explicit mention of uniformity, then Dlogtime-uniformity is intended. For uniform $\mathsf{NC}^1$ the situation is somewhat more complicated, as discussed in [Vol99]; there is wide agreement that the "correct" definition coincides with $\mathsf{ATIME}(O(\log n))$.

## 10.2 Hardness of MKTP under nonuniform many-one reductions

We modify the $\mathsf{ZPP}$ reduction of [AGvM+18, Section 3.1], which reduces the rigid graph isomorphism problem to MKTP, showing that it can be replaced by a nonuniform $\mathsf{AC}^0$ reduction. Here we say that a graph is *rigid* if it has no nontrivial automorphisms.

**Lemma 10.1.** *Let $A$ be any oracle. There is a function $f$ computable in Dlogtime-uniform $\mathsf{AC}^0$ such that, for any two rigid graphs $G_0, G_1$ with $n$ vertices:*

- $\Pr_r[f(G_0, G_1, r) \notin \mathrm{MKTP}^A] > 1 - \frac{1}{2^{4n^2}}$ *if* $G_0 \not\equiv G_1$, *and*
- $\Pr_r[f(G_0, G_1, r) \in \mathrm{MKTP}^A] = 1$ *if* $G_0 \equiv G_1$.

*Proof.* We present the proof for $A = \varnothing$; however, it is immediate that the proof carries over for any oracle $A$. The function $f$ is given by the reduction presented in [AGvM+18, Section 3.1], showing that the Rigid Graph Isomorphism Problem is in $\mathsf{Promise\text{-}ZPP}^{\mathsf{MKTP}}$. This reduction takes graphs $G_0$ and $G_1$ as input, and interprets the random coin flip sequence $r$ as a tuple $(w, \Pi)$ where $\Pi$ is a sequence of $t$ random permutations $\pi_1, \ldots, \pi_t$, and $|w| = t$.

We make use of a result of Hagerup [Hag91], showing that there is a function $e$ computed by Dlogtime-uniform $\mathsf{AC}^0$ circuits,[1] generating a nearly-uniform distribution on permutations of $n$ elements. More precisely, let $S_n$ denote the symmetric group on $[n]$, where permutation $\sigma$ is represented as a binary string of the form $\sigma(1) \ldots \sigma(n)$. Then for every $\ell$ there is a $k > \ell$ and a Dlogtime-uniform $\mathsf{AC}^0$-computable function $e : \{0,1\}^{n^k} \to S_n \cup \{0^{n \log n}\}$ such that, for every $\sigma \in S_n$

$$\Pr_{s \in \{0,1\}^{n^k}}[e(s) = \sigma] \geq 1/n! - 2^{-n^\ell}$$

and $\Pr_{s \in \{0,1\}^{n^k}}[e(s) = 0^{n \log n}] \leq 2^{-n^\ell}$.

Following the presentation in [AGvM+18], our $\mathsf{AC}^0$ reduction takes two graphs $G_0$ and $G_1$, along with a random string $r = w s_1 s_2 \ldots s_t$ where $|w| = t = n^{O(1)}$ and each $s_i$ has length $n^k$, where $k$ and $\ell$ (from the previous paragraph) are chosen so that $2^{-t/2} + t/2^{n^\ell} < 2^{-4n^2}$. Thus, for a randomly-chosen $r$, with probability at least $1 - (t/2^{n^\ell})$, in $\mathsf{AC}^0$ we can compute the pair $(w, \Pi)$ where $\Pi = \pi_1, \pi_2, \ldots \pi_t = e(s_1), e(s_2), \ldots, e(s_t)$. Next we compute the string $x_r =$

[1] Hagerup [Hag91] states this result in terms of CRCW PRAMs with a polynomial number of processors, running for $O(1)$ steps. It is known [BIS90] that this class coincides with Dlogtime-uniform $\mathsf{AC}^0$.

$\pi_1(G_{w_1}), \ldots, \pi_t(G_{w_t})$. (With probability at most $t/2^{n^\ell}$, some $e(s_i)$ consists of a block of zeros, in which case our $\mathsf{AC}^0$ function will set $x_r$ equal to a string of zeros, indicating failure.) We observe this is computable in $\mathsf{AC}^0$: Graphs are encoded as adjacency matrices. Thus, given a graph $G$ and a permutation $\pi$, the bit $(r, s)$ of $\pi(G)$ is the same as the bit $(i, j)$ in $G$, where $\pi(i) = r$ and $\pi(j) = s$. That is, position $(r, s)$ in the output is the $\mathrm{OR}_{i,j}$ (taken over all relevant positions $(i, j)$ in the encoding of $\pi$) of [$G_{i,j}$ AND [the encoding of $\pi$ contains the strings $(i, r)$ and $(j, s)$]]. This latter condition can easily be checked in $\mathsf{AC}^0$.

The proof in [AGvM$^+$18, Section 3.1] shows that, if $G_0 \equiv G_1$, then $\mathrm{KT}(x_r) \leq t(\log n!) + t/2$.

On the other hand, [AGvM$^+$18] observes that if $G_0 \not\equiv G_1$ then the entropy of the distribution on strings $x_r$ (assuming $t$ uniformly random permutations and a uniformly-randomly chosen string $w$) is at least $t + t\log(n!)$, and hence the probability that $\mathrm{KT}(x_r) < (t + t\log(n!)) - t/2$ is at most $2^{-t/2}$. In our setting, the permutations are *very nearly* uniformly random (and it approaches the uniform distribution as $\ell$ increases), and there is also the possibility that $x_r$ does not consist of $t$ permuted graphs, but instead is all zeros. This latter condition arises with probability at most $t/2^{n^\ell}$. Recalling that $2^{-t/2} + t/2^{n^\ell} < 2^{-4n^2}$, we now have the following:

- If $G_0 \equiv G_1$, then $\mathrm{KT}(x_r) \leq t(\log n!) + t/2$.
- If $G_0 \not\equiv G_1$, then with probability $> 1 - 2^{-4n^2}$, we have $\mathrm{KT}(x_r) \geq t(\log n!) + t/2$.

We are now ready to define the $\mathsf{AC}^0$-computable function $f$: $f(G_0, G_1, r) = (x_r, \theta)$, where $\theta = t(\log n!) + t/2$. We have established that $f$ has the desired properties. □

**Corollary 10.2.** *Let $A$ be any oracle. The rigid graph isomorphism problem is reducible to* $\mathrm{MKTP}^A$ *via a nonuniform* $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ *reduction.*

*Proof.* A standard counting argument shows that there is a value of $r$ that can be hardwired into the probabilistic reduction of Lemma 10.1 that works correctly for all pairs $(G_0, G_1)$ of $n$-vertex graphs. (Note that the length of the input is $2n^2$, and the error probability is at most $1/2^{4n^2}$.) □

**Theorem 10.3.** *Let $A$ be any oracle.* $\mathsf{DET}$ *is reducible to* $\mathrm{MKTP}^A$ *via a nonuniform* $\leq_{\mathrm{m}}^{\mathsf{NC}^0}$ *reduction.*

*Proof.* Since $\mathsf{DET}$ is closed under $\leq_{\mathrm{m}}^{\mathsf{TC}^0}$ reductions, it suffices to show that $\mathrm{MKTP}^A$ is hard under $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reductions, and then appeal to the "Gap" theorem of [AAR98], to obtain hardness under $\leq_{\mathrm{m}}^{\mathsf{NC}^0}$ reducibility. Torán [Tor04] shows that $\mathsf{DET}$ is $\mathsf{AC}^0$-reducible to GI. In fact it is shown in the proofs of Theorem 5.3 and Corollary 5.4 of [Tor04] that $\mathsf{DET}$ is $\mathsf{AC}^0$-reducible to GI via a reduction that produces only pairs of rigid graphs as output. Composing this reduction with the nonuniform $\mathsf{AC}^0$ reduction given by Corollary 10.2 completes the argument. □

We mention that the reduction of Theorem 10.3 is "natural" in the sense of [KC00]: A reduction is said to be *natural* if the size parameter of the reduction depends only on the input length.

Since our reduction is efficient, an appeal to the circuit lower bounds of Razborov and Smolensky [Raz87, Smo87] now yields the following corollary:

**Corollary 10.4.** $\mathrm{MKTP}^A$ *is not in* $\mathsf{AC}^0[p]$ *for any oracle* $A$ *and any prime* $p$.

In Chapter 12, we will strengthen Corollary 10.4 to an average-case lower bound.

## 10.3 Equivalence between hardness of MKTP and circuit lower bounds

The reader may wonder whether the nonuniform reduction can be made uniform under a suitable derandomization hypothesis. We do not know how to obtain a uniform $\mathsf{AC}^0$-many-one reduction, but we can come close, if the oracle $A$ is not too complex. Recall the definition of ctt-reductions: $B \leq_{\mathrm{ctt}}^{\mathcal{C}} C$ if there is a function $f \in \mathcal{C}$ with the property that $f(x)$ is a list $f(x) = (y_1, \ldots, y_m)$, and $x \in B$ if and only if $y_j \in C$ for all $j$. Furthermore, we say that $f$ is a *natural* logspace-uniform $\leq_{\mathrm{ctt}}^{\mathsf{AC}^0}$-reduction to MKTP if each query $y_j$ has the same length (and this length depends only on $|x|$), and furthermore each $y_j$ is of the form $(z_j, \theta)$ where the threshold $\theta$ depends only on $|x|$.

The following theorem can be viewed as a "partial converse" to results of [MW17, AHK17], which say that problems in $\mathsf{LTH} \subseteq \mathsf{E}$ require exponential size circuits if MCSP or MKTP is hard for $\mathsf{TC}^0$ under Dlogtime-uniform $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reductions.[2] That is, the earlier results show that very uniform hardness results imply circuit lower bounds, whereas the next theorem shows that somewhat stronger circuit lower bounds imply uniform hardness results (for a less-restrictive notion of uniformity, but hardness for a larger class). Later on, in Theorem 10.7, we present a related condition on reductions to $\mathrm{MKTP}^A$ that is *equivalent* to circuit lower bounds.

**Theorem 10.5.** *Let* $A$ *be any oracle. If there is some* $\epsilon > 0$ *such that* $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$*, then every language in* $\mathsf{DET}$ *reduces to* $\mathrm{MKTP}^A$ *via a natural logspace-uniform* $\leq_{\mathrm{ctt}}^{\mathsf{AC}^0}$*-reduction.*

*Proof.* Let $B \in \mathsf{DET}$. Thus there is an $\mathsf{AC}^0$ reduction $g$ reducing $B$ to the Rigid Graph Isomorphism Problem [Tor04]. Consider the following family of statistical tests $T_x(r)$, indexed by strings $x$:

> On input $r$:
> Compute $z = f(g(x), r)$, where $f(G_0, G_1, r)$ is the function from Lemma 10.1.
> Accept iff ($x \in B$ iff $z \in \mathrm{MKTP}^A$).

Since $B \in \mathsf{DET} \subseteq \mathsf{P}$, the test $T_x(r)$ has a polynomial-size circuit with one $\mathrm{MKTP}^A$ oracle gate. (In fact, the statistical test is an $\mathsf{NC}^2$ circuit with one oracle gate.) If $x \in B$, then $T_x$ accepts *every* string $r$, whereas if $x \notin B$, $T_x$ accepts most strings $r$.

Klivans and van Melkebeek [KvM02] (building on the work of Impagliazzo and Wigderson [IW97]) show that, if $\mathsf{DSPACE}(n)$ requires exponential-size circuits from a given class $\mathcal{C}$, then there is a hitting set generator computable in logspace that hits all large sets computable by circuits from $\mathcal{C}$ that have size $n^k$. In particular, under the given assumption, there is a function $h$ computable in logspace such that $h(0^n) = (r_1, r_2, \ldots, r_{n^c})$ with the property that, for all strings $x$ of length $n$, there is an element of $h(0^n)$ that is accepted by $T_x$.

[2] Recall that $\mathsf{LTH} = \bigcup_k \Sigma_k \mathsf{TIME}(O(n))$ is the linear-time analog of the polynomial time hierarchy.

Now consider the logspace-uniform $\mathsf{AC}^0$ oracle circuit family, where the circuit for inputs of length $n$ has the strings $h(0^n) = (r_1, r_2, \ldots, r_{n^c})$ hardwired into it. On input $x$, the circuit computes the queries $f(g(x), r_i)$ for $1 \leq i \leq n^c$, and accepts if, for all $i$, $f(g(x), r_i) \in \mathrm{MKTP}^A$. Note that if $x \notin B$, then one of the $r_i$ is accepted by $T_x$, which means that $f(g(x), r_i) \notin \mathrm{MKTP}^A$; if $x \in B$, then $f(g(x), r_i) \in \mathrm{MKTP}^A$ for all $i$. This establishes that the reduction is correct. □

It is also possible to prove a result analogous to Theorem 10.5, in terms of Dlogtime-uniform $\mathsf{AC}^0$ reductions, in place of logspace-uniform $\mathsf{AC}^0$ reductions. However, this requires a much stronger, *average case* circuit lower bound, for sets in $\mathsf{LTH}$ (as opposed to $\mathsf{DSPACE}(n)$):

**Theorem 10.6.** *Let $A$ be any oracle. There is a constant $c$ such that, if there is some $\epsilon > 0, b \geq 1$ and a set $B$ in the $d$-th level of $\mathsf{LTH}$ such that, for all large $n$ and every oracle circuit $C$ of size $2^{\epsilon n}$,*

$$\Pr_{x \in \{0,1\}^n}[B(x) = C^{\mathrm{MKTP}^A}(x)] < 1 - 1/n^b,$$

*then every language in $\mathsf{DET}$ reduces to $\mathrm{MKTP}^A$ via a natural Dlogtime-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$-reduction of depth $d + c$.*

*Proof.* The idea is similar to the proof of Theorem 10.5. Let $B \in \mathsf{DET}$. We consider the same family of statistical tests $T_x(r)$.

Viola [Vio05, Theorem 4.3] shows that, under the hypothesis of Theorem 10.6, there is a pseudorandom generator $G\colon \{0,1\}^{O(\log n)} \to \{0,1\}^{n^c}$ that is secure against all statistical tests computable by circuits of size $n^c$. In particular, as in the proof of Theorem 10.5, we obtain a hitting set generator $h$. The depth required by the construction in [Vio05] is $d + O(1)$.

The rest of the proof proceeds precisely as in the proof of Theorem 10.5. □

We remark that the hardness assumption of Theorem 10.5 ($\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$) can probably be weakened (saying that $\mathsf{DSPACE}(n)$ requires large circuits of some restricted sort), since the class of statistical tests that need to be fooled consists only of $\mathsf{NC}^2$ circuits with one oracle gate. On the other hand, Theorem 10.7 indicates that the hardness assumption that we use is *equivalent* to the existence of uniform reductions, for certain oracles $A$ – so it is not clear that there is much to be gained by searching for a weaker hardness assumption.

Theorem 10.5 deals with the oracle problem $\mathrm{MKTP}^A$, but the most interesting case is the case where $A = \varnothing$, both because the hypothesis seems most plausible in that case, and because MKTP has been studied in connection with MCSP, which has been studied more than the associated oracle circuit problem $\mathrm{MCSP}^A$. The hypothesis is false when $A = \mathsf{QBF}$, since the $\mathrm{KT}^A$ measure is essentially the same as the KS measure studied in [ABK+06b], where it is shown that $\mathsf{PSPACE} = \mathsf{ZPP}^{R_{\mathrm{KS}}}$, and thus $\mathsf{PSPACE}$ has polynomial-size $\mathrm{MKTP}^{\mathsf{QBF}}$-circuits. Strikingly, it is of interest that not only the hypothesis is false in this case – but the conclusion is false as well. (See Corollary 10.10.)

For certain oracles (and we discuss below how broad this class of oracles is), the existence of uniform reductions is *equivalent to* certain circuit lower bounds.

**Theorem 10.7.** *Let $\mathrm{MKTP}^A \in \mathsf{P}^A/\mathsf{poly}$. Then the following are equivalent:*

- $\mathsf{PARITY}$ *reduces to $\mathrm{MKTP}^A$ via a natural logspace-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$-reduction.*

- *For some* $\epsilon > 0$, $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^A(2^{\epsilon n})$.
- *For some* $\epsilon > 0$, $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$.
- $\mathsf{DET}$ *reduces to* $\mathrm{MKTP}^A$ *via a natural logspace-uniform* $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$*-reduction.*

*Furthermore, if* $\mathsf{PARITY}$ *reduces to* $\mathrm{MCSP}^A$ *via a natural logspace-uniform* $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$*-reduction, then all of the above hold.*

*Proof.* First, we show that the first condition implies the second.

Let $\{C_n : n \in \mathbb{N}\}$ be a logspace-uniform family of oracle circuits computing $\mathsf{PARITY}$, consisting of $\mathsf{AC}^0$ circuitry feeding into oracle gates, which in turn are connected to an AND gate as the output gate. Let the oracle gates in $C_n$ be $g_1, g_2, \ldots, g_{n^c}$. On any input string $x$, let the value fed into gate $g_i$ on input $x$ be $(q_i(x), \theta)$, and recall that, since the reduction is natural, the threshold $\theta$ depends only on $n$, and thus it is a constant in $C_n$.

At this point, it is useful to recall a lemma from [AHK17] (distilled from [MW17]) that describes how the complexity depends on $\theta$:

**Lemma 10.8.** *[AHK17, Claim 3.11] For any language* $A$ *and any* $0 \leq v \leq m$, $\mathrm{MCSP}^A$ *on inputs* $f \in \{0,1\}^m$*, with the size parameter fixed to* $\theta$*, is solved by a DNF formula of size* $m \cdot 2^{O(\theta^2 \log \theta)}$.

Thus, by Lemma 10.8, each $\mathrm{MKTP}^{\mathsf{QBF}}$ oracle gate can be replaced by a DNF formula of size at most $n^{O(1)}2^{O(\theta^2 \log \theta)}$. Inserting these DNF formulae into $C_n$ (in place of each oracle gate) results in a circuit of size $n^{O(1)}2^{O(\theta^2 \log \theta)}$ computing $\mathsf{PARITY}$. Let the depth of this circuit be some constant $d$. It follows from [Hås86] that $n^{O(1)}2^{O(\theta^2 \log \theta)} \geq 2^{\Omega(n^{1/(d-1)})}$, and hence that $\theta \geq n^{1/4d}$.

Note that all of the oracle gates $g_i$ must output 1 on input $0^{n-1}1$, and one of the oracle gates $g_{i_0}$ must output 0 on input $0^n$. Thus we have $\mathrm{KT}^A(q_{i_0}(0^n)) \geq \theta \geq n^{1/4d}$. It follows from [ABK+06b, Theorem 11] that the function with truth table $q_{i_0}(0^n)$ has no circuit (with oracle gates for $A$) of size less than $(\mathrm{KT}^A(q_{i_0}(0^n)))^{1/3} \geq \theta^{1/3} \geq n^{1/12d}$.

Note that, in order to compute the $j$-th bit of some query $q_i(0^n)$, it suffices to evaluate a logspace-uniform $\mathsf{AC}^0$ circuit where all of the input bits are 0. Since this computation can be done in logspace on input $(0^n 1^i 0^j)$, note that the language $H = \{(n,i,j) :$ the $j$-th bit of query $q_i(0^n)$ is $1\}$ is in linear space. Let $m = |(n,i,j)|$, and let $s(m)$ be the size of the smallest circuit $D_m$ computing $H$ for inputs of length $m$. Hardwire the bits for $n$ and also set the bits for $i$ to $i_0$. The resulting circuit on $|j| < m$ bits computes the function given by $q_{i_0}(0^n)$, and it was observed above that this circuit has size at least $n^{1/20d} \geq 2^{m/20d}$.

This establishes the first implication. (Note also that a similar argument yields the same conclusion from the assumption that $\mathsf{PARITY}$ reduces to $\mathrm{MCSP}^A$ via a natural logspace-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$-reduction.)

The assumption that $\mathrm{MKTP}^A \in \mathsf{P}^A/\mathsf{poly}$ suffices to show that the second condition implies the third. More formally, we'll consider the contrapositive. Assume that $\mathsf{DSPACE}(n) \subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$ for every $\epsilon > 0$. An oracle gate for $\mathrm{MKTP}^A$ on inputs of size $m$ can be replaced by a circuit (with oracle gates for $A$) of size $m^c$ for some constant $c$. Carrying out this substitution in a circuit (with oracle gates for $\mathrm{MKTP}^A$) of size $2^{\epsilon n}$ yields a circuit of size at most $2^{\epsilon n} + 2^{\epsilon n}(2^{\epsilon n})^c$.

Let $\delta > 0$. Then we can pick $\epsilon$ small enough so that $2^{\epsilon n} + 2^{\epsilon n}(2^{\epsilon n})^c < 2^{\delta n}$, thereby establishing that $\mathsf{DSPACE}(n) \subseteq \mathsf{i.o.SIZE}^A(2^{\delta n})$ for every $\delta > 0$. This establishes the second implication.

Theorem 10.5 establishes that the third condition implies the fourth. The fourth condition obviously implies the first. □

To the best of our knowledge, this is the first theorem that has given conditions where the existence of a reduction to $\mathrm{MCSP}^A$ implies the existence of a reduction to $\mathrm{MKTP}^A$. We know of no instance where the implication goes in the opposite direction.

The logspace uniformity condition in Theorem 10.7 can be replaced by other less-restrictive uniformity conditions. We mention the following example:

**Corollary 10.9.** *Let* $\mathrm{MKTP}^A \in \mathsf{P}^A/\mathsf{poly}$. *Then the following are equivalent:*

- $\mathsf{PARITY}$ *reduces to* $\mathrm{MKTP}^A$ *via a natural* $\mathsf{P}$*-uniform* $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$*-reduction.*
- *For some* $\epsilon > 0$, $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}^A(2^{\epsilon n})$.
- *For some* $\epsilon > 0$, $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$.
- $\mathsf{DET}$ *reduces to* $\mathrm{MKTP}^A$ *via a natural* $\mathsf{P}$*-uniform* $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$*-reduction.*

*Furthermore, if* $\mathsf{PARITY}$ *reduces to* $\mathrm{MCSP}^A$ *via a natural* $\mathsf{P}$*-uniform* $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$*-reduction, then all of the above hold.*

At this point, we should consider the class of oracles for which Theorem 10.7 applies. That is, what is the set of oracles $A$ for which $\mathrm{MKTP}^A \in \mathsf{P}^A/\mathsf{poly}$? First, we observe that this condition holds for any $\mathsf{PSPACE}$-complete set, which yields the following corollary:

**Corollary 10.10.** $\mathsf{PARITY}$ *does not reduce to either* $\mathrm{MKTP}^{\mathsf{QBF}}$ *or* $\mathrm{MCSP}^{\mathsf{QBF}}$ *via a natural logspace-uniform* $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$*-reduction.*

**Remark.** As an instructive example of an oracle $A$ for which $\mathrm{MKTP}^A \in \mathsf{P}^A/\mathsf{poly}$, consider the set $A = \{(M, x, 1^m) : M$ is an alternating Turing machine that accepts $x$, and runs in time at most $m$ and makes at most $\log m$ alternations$\}$. $A$ is complete for the class $\mathsf{ATIME} - \mathsf{ALT}(n^{O(1)}, O(\log n))$ under $\leq^{\mathsf{AC}^0}_{\mathrm{m}}$ reductions. It is easy to see that $\mathrm{MKTP}^A \in \mathsf{ATIME} - \mathsf{ALT}(n^{O(1)}, O(\log n))$, and thus $\mathrm{MKTP}^A \in \mathsf{P}^A$. (Other examples can easily be created in this way, using an even smaller number of alternations.) Note that, for this oracle $A$, it seems plausible that all four conditions in Theorem 10.7 hold.

Nonetheless, we do grant that this does seem to be a strong condition to place upon the oracle $A$ – and it has even stronger consequences than are listed in Theorem 10.7. For instance, note that the proof that the first condition in Theorem 10.7 implies the second relies only on the fact that $\mathsf{PARITY}$ requires large $\mathsf{AC}^0$ circuits. Thus, an identical proof shows that these four conditions are also equivalent to the condition that $\mathsf{PARITY}$ is reducible to $\mathrm{MKTP}^A$ via a natural ctt-reduction where the queries are computed by logspace-uniform $\mathsf{AC}^0[7]$ circuits. (Or you can substitute any other problem and class of mod circuits, where an exponential lower bound is known because of [Raz87, Smo87].) In fact, as in the proof of [AHK17, Lemma 3.10] we can apply random restrictions in a logspace-uniform way (as described in [Agr11]) and obtain a reduction from $\mathsf{PARITY}$ to $\mathrm{MKTP}^A$ where the queries are computed by logspace-uniform $\mathsf{NC}^0$ circuits! For example, here is an argument showing that $\mathsf{MAJORITY}$ is reducible to $\mathrm{MKTP}^A$ (for oracle $A$ satisfying the hypotheses of Theorem 10.7) via natural ctt-reductions

computed by logspace-uniform $\mathsf{AC}^0[3]$ circuits iff PARITY is reducible to $\mathrm{MKTP}^A$ via reductions where the queries are computed by logspace-uniform $\mathsf{NC}^0$ circuits:

Assume first that MAJORITY is reducible to $\mathrm{MKTP}^A$ via natural ctt-reductions computed by logspace-uniform $\mathsf{AC}^0[3]$ circuits. The proof that the first condition in Theorem 10.7 implies the second also shows that the second condition holds if MAJORITY is reducible to $\mathrm{MKTP}^A$ via natural ctt-reductions computed by logspace-uniform $\mathsf{AC}^0[3]$ circuits. (The only things that need to be changed, are (1) every occurrence of "PARITY" should be changed to MAJORITY" (2) the phrase "consisting of $\mathsf{AC}^0$ circuitry feeding into oracle gates" should be changed to "consisting of $\mathsf{AC}^0[3]$ circuitry feeding into oracle gates", and (3) "Note that all of the oracle gates $g_i$ must output 1 on input $0^{n-1}1$" should be replaced by "Note that all of the oracle gates $g_i$ must output 1 on input $1^n$".) Thus, under our assumption, all four of the conditions in Theorem 10.7 hold. In particular, PARITY reduces to $\mathrm{MKTP}^A$ via a natural logspace-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$-reduction. The $\mathsf{AC}^0$-computable function $f$ that computes the list of oracle queries has the property that there is a logspace-computable restriction $\rho$ that leaves $n^\epsilon$ input variables unset (for some $\epsilon > 0$) with the property that the function $f|_\rho$ on $n^\epsilon$ variables is $\mathsf{NC}^0$-computable. (See, e.g., [AAR98, Lemma 7] and see [Agr11] to see how $\rho$ can be computed in logspace.) This yields the claimed reduction from PARITY to $\mathrm{MKTP}^A$ where the queries are $\mathsf{NC}^0$-computable.

Conversely, assume now that PARITY is reducible to $\mathrm{MKTP}^A$ via a natural $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$-reduction where the queries are computed by logspace-uniform $\mathsf{NC}^0$ circuits. This is a stronger condition than the first condition on Theorem 10.7, and hence all four of these conditions hold. In particular, DET reduces to $\mathrm{MKTP}^A$ via $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$ reductions. The claim now follows, since MAJORITY $\in$ DET.

We find these implications to be surprising. The "gap" phenomenon that was described in [AAR98] (showing that completeness under one class of reductions is equivalent to completeness under a more restrictive class of reductions) had not previously been observed to apply to $\mathsf{AC}^0[p]$ reducibility.

We want to highlight some contrasts between Theorem 10.5 and Corollary 10.10. $\mathrm{MKTP}^{\mathsf{QBF}}$ is hard for PSPACE under ZPP-Turing reductions [ABK+06b], whereas MKTP is in NP. Thus $\mathrm{MKTP}^{\mathsf{QBF}}$ appears to be *much harder* than MKTP. Yet, Theorem 10.5 shows that, under a plausible hypothesis, the "easier" set MKTP is hard for DET, whereas (by Corollary 10.10) the "harder" problem $\mathrm{MKTP}^{\mathsf{QBF}}$ cannot even be used as an oracle for PARITY under this same reducibility. In other words, the (conditional) natural logspace-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$ reductions from problems in DET to MKTP given in Theorem 10.5 are not "oracle-independent" in the sense of Chapter 7.

Prior to this work, it appears that there was no evidence for any variant of MCSP or MKTP being hard for a reasonable complexity class under $\leq^{\mathsf{L}}_{\mathrm{T}}$ reductions. All prior reductions (such as those in [AD17, ABK+06b, AGvM+18]) had been probabilistic and/or nonuniform, or (even under derandomization hypotheses) seemed difficult to implement in NC. But Theorem 10.7 shows that it is quite likely that MKTP is hard for DET under $\leq^{\mathsf{L}}_{\mathrm{T}}$ reductions (and even under much more restrictive reductions). Previously, we had viewed the results of [AHK17] as providing evidence that none of these variants would be hard for P under, say, logspace reducibility. Now, we are no longer sure what to expect.

## 10.4 On the importance of uniformity

Surprisingly, the notion of uniformity appears to be central. In particular, the reader is probably wondering whether the logspace-uniformity condition in Theorem 10.5 (relating hardness of $\mathrm{MKTP}^A$ to worst-case circuit lower bounds) can be improved to Dlogtime-uniformity. As a partial answer to this question, we note that Viola [Vio05] shows that there is no black-box construction of a pseudorandom generator computable in $\mathsf{AC}^0$ that is based on worst-case circuit lower bounds. In this section, in Theorem 10.12, we show that, when considering hardness of MKTP and MCSP, small details about the complexity of the reduction (including the precise depth, and the notion of uniformity) cannot be ignored.

First, we recall Corollary 3.7 of [AHK17], which states that $\mathrm{MKTP}^{\mathsf{QBF}}$ is not hard for $\mathsf{P}$ under $\leq^{\mathsf{L}}_{\mathrm{m}}$ reductions unless $\mathsf{PSPACE} = \mathsf{EXP}$. It turns out that this holds even for logspace-Turing reductions.

**Theorem 10.11.** $\mathrm{MKTP}^{\mathsf{QBF}}$ *is not hard for* $\mathsf{P}$ *(or* $\mathsf{NP}$*) under* $\leq^{\mathsf{L}}_{\mathrm{T}}$ *reductions unless* $\mathsf{PSPACE} = \mathsf{EXP}$ *(*$\mathsf{PSPACE} = \mathsf{NEXP}$*, respectively).* $\mathrm{MKTP}^{\mathsf{QBF}}$ *is not hard for* $\mathsf{PSPACE}$ *under* $\leq^{\mathsf{L}}_{\mathrm{T}}$ *reductions. The same holds for* $\mathrm{MCSP}^{\mathsf{QBF}}$*.*

We include this proof here, both because it improves a corollary in [AHK17], and because the proof can be viewed as a warm-up for the proof of Theorem 10.12.

*Proof.* First, note that $\leq^{\mathsf{L}}_{\mathrm{T}}$ and $\leq^{\mathsf{L}}_{\mathrm{tt}}$ reducibilities coincide [LL76]. Thus assume that $\mathrm{MKTP}^{\mathsf{QBF}}$ is hard for $\mathsf{P}$ under $\leq^{\mathsf{L}}_{\mathrm{tt}}$ reductions; we will show that $\mathsf{PSPACE} = \mathsf{EXP}$. (The proof for $\mathrm{MCSP}^{\mathsf{QBF}}$ is identical, and the variant concerning hardness for $\mathsf{NP}$ is analogous.)

The proof idea is as follows: Assume that $\mathsf{P} \subseteq \mathsf{L}_{\mathrm{tt}}^{\mathrm{MKTP}^{\mathsf{QBF}}}$. (Here, $\mathsf{L}_{\mathrm{tt}}$ means a $\leq^{\mathsf{L}}_{\mathrm{tt}}$ reduction.) By standard padding, we obtain $\mathsf{EXP} \subseteq \mathsf{PSPACE}_{\mathrm{tt}}^{\mathrm{MKTP}^{\mathsf{QBF}}}$. Any query of a $\mathsf{PSPACE}_{\mathrm{tt}}$ machine has low $\mathrm{KT}^{\mathsf{QBF}}$ complexity. Moreover, one can check whether a string has low $\mathrm{KT}^{\mathsf{QBF}}$ complexity in $\mathsf{PSPACE}$. Combining these two facts, we obtain $\mathsf{EXP} \subseteq \mathsf{PSPACE}_{\mathrm{tt}}^{\mathrm{MKTP}^{\mathsf{QBF}}} = \mathsf{PSPACE}$. A formal proof follows.

Let $B \in \mathsf{EXP}$. Let $B' = \{x10^{2^{|x|}} : x \in B\}$ and note that $B' \in \mathsf{P}$. Consider the $\leq^{\mathsf{L}}_{\mathrm{tt}}$ reduction that reduces $B'$ to $\mathrm{MKTP}^A$. On any input string $y$, let the $i$-th oracle query be $q_i(y)$. The language $\{(i, j, x) :$ the $j$-th bit of $q_i(x10^{2^{|x|}})$ is $1\}$ is in $\mathsf{PSPACE}$ and thus is in $\mathsf{P}^{\mathsf{QBF}}$. It follows that $q_i(x10^{2^{|x|}})$ is of the form $(y_i, \theta_i)$, where $\mathrm{KT}^{\mathsf{QBF}}(y_i) = |x, i, j|^{O(1)}$. Thus, to evaluate the oracle query $q_i$ on input $x10^{2^{|x|}}$, this $\mathsf{PSPACE}$ computation (on input $x$) suffices: Compute the bits of $\theta_i$; this can be done in $\mathsf{PSPACE}$, since the number of bits in $\theta_i$ is at most $|x|^{O(1)}$, and each bit is computable in $\mathsf{PSPACE}$. If $\theta_i > |x, i, j|^c$ (for the appropriate value of $c$), then return "1" since the query $y_i$ certainly has $\mathrm{KT}^A$ complexity less than this. Otherwise, try all descriptions $d$ of length at most $\theta_i$, to determine whether there is some such $d$ for which $U^{\mathsf{QBF}}(d, j)$ is equal to the $j$-th bit of $q_i$ (allowing at most $|x, i, j|^c$ steps for the computation of $U$).

The rest of the $\leq^{\mathsf{L}}_{\mathrm{tt}}$ reduction on input $x10^{2^{|x|}}$ can be computed in space $|x|^{O(1)}$, by re-computing the values of the oracle queries, as required.

The unconditional result that $\mathrm{MKTP}^{\mathsf{QBF}}$ is not hard for $\mathsf{PSPACE}$ under $\leq^{\mathsf{L}}_{\mathrm{T}}$ reductions follows along the same lines, choosing $B \in \mathsf{EXPSPACE}$, and leading to the contradiction $\mathsf{EXPSPACE} = \mathsf{PSPACE}$. □

A similar approach yields the following result:

**Theorem 10.12.** *For each $d \geq 0$, if $\Sigma^p_{d+2} \subseteq \mathsf{P}^A/\mathsf{poly}$ and $\mathsf{PSPACE} \not\subseteq \mathsf{PH}^A$, then neither $\mathrm{MKTP}^A$ nor $\mathrm{MCSP}^A$ is hard for $\mathsf{NC}^1$ under Dlogtime-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{tt}}$ reductions of depth $d$.*

*Proof.* We present the proof for $\mathrm{MKTP}^A$; the proof for $\mathrm{MCSP}^A$ is identical.

Assume that $\mathrm{MKTP}^A$ is hard for $\mathsf{NC}^1$ under Dlogtime-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{tt}}$ reductions of depth $d$; we will show that $\mathsf{PSPACE} \subseteq \mathsf{PH}^A$.

By the closure properties of $\mathsf{PH}$, it will suffice to show that $\mathsf{ATIME}(n) \subseteq \mathsf{PH}^A$.

Let $B \in \mathsf{ATIME}(n)$. Let $B' = \{x10^{2^{|x|}} : x \in B\}$ and note that $B' \in \mathsf{NC}^1$. Consider the oracle family $(C_m)$ that reduces $B'$ to $\mathrm{MKTP}^A$. Let the oracle gates in $C_{2^n+n+1}$ be $g_1, g_2, \ldots, g_\ell$. On any input string $y$, let the query that is fed into gate $g_i$ be $q_i(y)$. The language $\{(2^{|x|} + |x| + 1, i, j, x) :$ the $j$-th bit of $q_i(x10^{2^{|x|}})$ is $1\}$ is in $\Sigma^p_{d+2}$ and thus is in $\mathsf{P}^A/\mathsf{poly}$. It follows that $q_i(x10^{2^{|x|}})$ is of the form $(y_i, \theta_i)$, where $\mathrm{KT}^A(y_i) = |x, i, j|^{O(1)}$. Thus, to evaluate oracle gate $g_i$ on input $x10^{2^{|x|}}$, this $\mathsf{PH}^A$ computation (on input $x$) suffices: Compute the bits of $\theta_i$; this can be done in $\mathsf{PH}$, since the number of bits in $\theta_i$ is at most $|x|^{O(1)}$, and each bit is computable in $\mathsf{PH}$. If $\theta_i > |x, i, j|^c$ (for the appropriate value of $c$), then return "1" since the query $y_i$ certainly has $\mathrm{KT}^A$ complexity less than this. Otherwise, guess a description $d$ of length at most $\theta_i$, and universally check (for each $j$) that $U^A(d, j)$ is equal to the $j$-th bit of $q_i$ (allowing at most $|x, i, j|^c$ steps for the computation of $U$).

To evaluate the rest of the circuit, note that the unbounded fan-in AND and OR gates that sit just above the oracle gates can also be evaluated in $\mathsf{PH}^A$ (at one level higher in the hierarchy than is required to evaluate the oracle gates). Repeating this process through the remaining $O(1)$ levels of the circuit yields the desired $\mathsf{PH}^A$ algorithm for $B$. □

**Remark.** The significance of Theorem 10.12 is best viewed by combining it with Theorem 10.5. If we choose $A$ to be any $\mathsf{PP}$-complete set, or if we choose $A$ to be one of the sets discussed in the Remark after Corollary 10.10, then for all $d$ we have $\Sigma^p_{d+2} \subseteq \mathsf{P}^A$ and both of the hypotheses

- $\mathsf{PSPACE} \not\subseteq \mathsf{PH}^A$, and
- $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^{\mathrm{MKTP}^A}(2^{\epsilon n})$

are plausible. Thus, for such oracles $A$, under a plausible hypothesis, we have both $\mathrm{MKTP}^A$ is *not* hard for $\mathsf{NC}^1$ under Dlogtime-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{tt}}$ reductions, and $\mathrm{MKTP}^A$ *is* hard for $\mathsf{DET}$ under logspace-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$ reductions. Thus different notions of uniformity are a key part of the puzzle, when trying to understand the hardness of problems such as MKTP and MCSP.

As another example, choose $A$ to be any set that is complete for $\Sigma^p_{d+2}$, and assume $\mathsf{PSPACE} \neq \mathsf{PH}$. Then under the strong-but-plausible hypothesis that there is a set $B \in \Sigma_{d'}\mathsf{TIME}(n)$ that has large symmetric difference with any set in $\mathsf{SIZE}^{\mathrm{MKTP}^A}(2^{\epsilon n})$, we have $\Sigma^p_{d+2} \subseteq \mathsf{P}^A$ and $\mathsf{PSPACE} \not\subseteq \mathsf{PH}^A = \mathsf{PH}$, thereby satisfying the hypotheses of both Theorem 10.12 and Theorem 10.6. Thus, for this choice of $A$, under a plausible hypothesis, we have both $\mathrm{MKTP}^A$ is *not* hard for $\mathsf{NC}^1$ under Dlogtime-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{tt}}$ reductions of depth $d$, and $\mathrm{MKTP}^A$ *is* hard for $\mathsf{DET}$ under Dlogtime-uniform $\leq^{\mathsf{AC}^0}_{\mathrm{ctt}}$ reductions of depth $d' + c$ (where $c$ is the constant from Theorem 10.6).

In both of these examples, the key ingredient seems to be that, in order for $\mathsf{AC}^0$ to be able to reduce problems to $\mathrm{MCSP}^A$ or $\mathrm{MKTP}^A$, it is essential to be

able to formulate useful queries, by either having sufficient depth, or by having sufficient power in the uniformity condition.

We are even able to extend our approach in some cases, to apply to $\mathsf{AC}^0$-Turing reducibility.

**Theorem 10.13.** *Let* $\mathsf{NP}^A \subseteq \mathsf{P}^A/\mathsf{poly}$. *If* $\mathsf{PSPACE} \not\subseteq \mathsf{PH}^A$, *then neither* $\mathrm{MKTP}^A$ *nor* $\mathrm{MCSP}^A$ *is hard for* $\mathsf{NC}^1$ *under Dlogtime-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions.*

*Proof.* The proof is similar to that of Theorem 10.12. Assume that $\mathrm{MKTP}^A$ is hard for $\mathsf{NC}^1$ under Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions; we will show that $\mathsf{ATIME}(n) \subseteq \mathsf{PH}^A$ by presenting a $\mathsf{PH}^A$ algorithm to evaluate the gates in the $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reduction of the $\mathsf{NC}^1$ language $B'$ from the proof of Theorem 10.12.

Note that in a circuit computing an $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reduction, there is an "initial" layer of oracle gates, whose queries are computed nonadaptively, while all oracle gates at deeper levels have queries whose values depend upon oracle gates at earlier levels in the circuit. Note also that, under the given assumption $\mathsf{NP}^A \subseteq \mathsf{P}^A/\mathsf{poly}$, we can conclude that $\mathsf{PH}^A \subseteq \mathsf{P}^A/\mathsf{poly}$.

The proof now proceeds along precisely the same lines as the proof of Theorem 10.12, which shows that a $\mathsf{PH}^A$ computation can compute the value of each wire that feeds into the "initial" layer of oracle gates. Similarly, as in the proof of Theorem 10.12, all of the AND, OR, and NOT gates at higher levels can be computed in $\mathsf{PH}^A$, given that the gates at lower levels can be evaluated in $\mathsf{PH}^A$. Thus, we need only show how to deal with oracle gates at deeper levels.

Consider any such oracle gate $g$. On any input string $y$, let the query that is fed into gate $g$ when evaluating the circuit on input $y$ be $q_g(y)$. The language $\{(2^{|x|}+|x|+1, g, j, x) : \text{the } j\text{-th bit of } q_g(x10^{2^{|x|}}) \text{ is } 1\}$ is in $\mathsf{PH}^A$ and thus (by our new assumption) is in $\mathsf{P}^A/\mathsf{poly}$. It follows that $q_g(x10^{2^{|x|}})$ is of the form $(y, \theta)$, where $\mathrm{KT}^A(y) = |x, g, j|^{O(1)}$. Thus, to evaluate oracle gate $g$ on input $x10^{2^{|x|}}$, this $\mathsf{PH}^A$ computation (on input $x$) suffices: Compute the bits of $\theta$; this can be done in $\mathsf{PH}^A$, since the number of bits in $\theta_i$ is at most $|x|^{O(1)}$, and each bit is computable in $\mathsf{PH}$. If $\theta_i > |x, g, j|^c$ (for the appropriate value of $c$), then return "1" since the query $y$ certainly has $\mathrm{KT}^A$ complexity less than this. Otherwise, guess a description $d$ of length at most $\theta$, and universally check (for each $j$) that $U^A(d, j)$ is equal to the $j$-th bit of $q_g$ (allowing at most $|x, i, j|^c$ steps for the computation of $U$). □

In order to compare our results with those of [AHK17, MW17], we also state a related theorem, whose proof is similar:

**Theorem 10.14.** *Let* $\mathsf{NP}^A \subseteq \mathsf{P}^A/\mathsf{poly}$. *If* $\mathsf{CH} \not\subseteq \mathsf{PH}^A$, *then neither* $\mathrm{MKTP}^A$ *nor* $\mathrm{MCSP}^A$ *is hard for* $\mathsf{TC}^0$ *under Dlogtime-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions.*

*Proof.* The proof is nearly identical to that of Theorem 10.13.

Under the assumption that $\mathrm{MKTP}^A$ is hard for $\mathsf{TC}^0$ under Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions; it suffices to show that the linear-time counting hierarchy (see [AKR+01] for a definition) is contained in $\mathsf{PH}^A$ by presenting a $\mathsf{PH}^A$ algorithm for a set $B$ in the linear-time counting hierarchy. The language $B'$ is now in $\mathsf{TC}^0$, instead of merely being in $\mathsf{NC}^1$. The rest of the proof proceeds virtually unchanged. (One can modify the statement of Theorem 10.12 in a similar way, but we do not include this modification here.) □

A consequence of Theorem 10.13 and Theorem 10.14 is the following corollary, which has the same flavor of results of the form "MCSP is hard for class $\mathcal{C}$ implies a likely but hard-to-prove consequence" as presented by Murray and Williams [MW17], but moving beyond the $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reductions considered by them, to the more general $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions.

**Corollary 10.15.** *If either of* MKTP *or* MCSP *is hard for* $\mathsf{NC}^1$ *(or* $\mathsf{TC}^0$*) under Dlogtime-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions, then* $\mathsf{NP} \neq \mathsf{NC}$ *(*$\mathsf{NP} \neq \mathsf{TC}^0$*, respectively).*

*Proof.* This follows from Theorem 10.13 and Theorem 10.14 when $A = \varnothing$. If $\mathsf{NP} = \mathsf{NC}$, then $\mathsf{NP} \subseteq \mathsf{P/poly}$, and $\mathsf{PH} = \mathsf{NC} \neq \mathsf{PSPACE}$. Thus neither MKTP nor MCSP is hard for $\mathsf{NC}^1$ under Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions. Also, if $\mathsf{NP} = \mathsf{TC}^0$, then $\mathsf{NP} \subseteq \mathsf{P/poly}$, and $\mathsf{PH} = \mathsf{TC}^0 \neq \mathsf{CH}$ [All99]. Thus neither MKTP nor MCSP is hard for $\mathsf{TC}^0$ under Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions. □

Corollary 10.15 should be compared to the earlier work of [MW17, AHK17]. Murray and Williams presented nonuniform lower bounds that would follow from MCSP or MKTP being hard for NP under Dlogtime-uniform $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reductions. In [AHK17] even stronger nonuniform consequences were shown to follow from the weaker assumption of hardness for $\mathsf{TC}^0$. (See Table 10.2.) In Theorem 10.13, we present a weaker *uniform* lower bound that follows from the weaker assumption that MCSP or MKTP is hard for $\mathsf{TC}^0$ under a *more powerful* notion of reducibility.

Table 10.2: Consequences of hardness for MCSP and MKTP: If MCSP or MKTP is $\mathcal{C}$-hard under $\mathcal{R}$, then condition $\mathcal{S}$ holds. The last column shows where the result is found.

| class $\mathcal{C}$ | reductions $\mathcal{R}$ | statement $\mathcal{S}$ | Reference |
|---|---|---|---|
| $\mathsf{TC}^0$ | Dlogtime-uniform $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ | $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ and $\mathsf{LTH} \not\subseteq \mathsf{i.o.SIZE}[2^{\epsilon n}]$ | [AHK17] |
| PARITY | $\mathsf{L}$-uniform $\leq_{\mathrm{ctt}}^{\mathsf{AC}^0}$ | MKTP $\notin \mathsf{P/poly}$ or $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}[2^{\epsilon n}]$ | Theorem 10.7 |
| $\mathsf{TC}^0$ | Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ | $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{CH} = \mathsf{PH}$ (hence $\mathsf{NP} \neq \mathsf{TC}^0$) | Corollary 10.15 |
| $\mathsf{NC}^1$ | Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ | $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{PSPACE} = \mathsf{PH}$ (hence $\mathsf{NP} \neq \mathsf{NC}$) | Corollary 10.15 |
| NP | Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ | $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{NEXP} = \mathsf{MA}$ (hence $\mathsf{NP} \neq \mathsf{MA} \cap \mathsf{P/poly}$) | Corollary 10.16 |
| NP | $\mathsf{L}$-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ | $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{NEXP} = \mathsf{PSPACE}$ | Corollary 10.19 |
| NP | P-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ | $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{NEXP} = \mathsf{EXP}$ | Corollary 10.19 |

We also present another result in this vein, about NP-completeness. Prior work [MW17, AHK17] had obtained stronger consequences from the stronger assumption that MCSP is NP-complete under Dlogtime-uniform $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reductions.

**Corollary 10.16.** *If either of* MKTP *or* MCSP *is hard for* NP *under Dlogtime-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions, then*

$$\mathsf{NP} \neq \mathsf{MA} \cap \mathsf{P/poly}.$$

*Proof.* If you modify the proof of Theorem 10.13, replacing $\mathsf{NC}^1$ by NP and replacing PSPACE by NEXP, you obtain that, if $\mathsf{NP} \subseteq \mathsf{P/poly}$, then $\mathsf{NEXP} \neq \mathsf{PH}$ implies that neither MKTP nor MCSP is hard for NP under Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions. (That is, if we assume that MKTP is hard for NP under Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions, then the argument from Theorem 10.13 shows that

$\mathsf{NEXP} \subseteq \mathsf{PH}$, by presenting a $\mathsf{PH}$ algorithm to evaluate the gates in an $\mathsf{AC}^0$ oracle circuit reducing an $\mathsf{NP}$ language $B'$ to MKTP.)

Or, restating this using the same hypothesis as in the statement of the corollary, if MKTP or MCSP is hard for $\mathsf{NP}$ under Dlogtime-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$, then either $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{NEXP} = \mathsf{PH}$. Since ($\mathsf{NP} \subseteq \mathsf{P/poly}$ and $\mathsf{NEXP} = \mathsf{PH}$) is equivalent to $\mathsf{NEXP} \subseteq \mathsf{P/poly}$, and since $\mathsf{NEXP} \subseteq \mathsf{P/poly}$ is equivalent to $\mathsf{NEXP} = \mathsf{MA}$ [IKW02], we obtain that $\mathsf{NP}$-hardness of MCSP or MKTP implies $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ or $\mathsf{NEXP} = \mathsf{MA}$. (Murray and Williams obtain essentially this same consequence under the stronger assumption that MCSP is complete under $\leq_{\mathrm{m}}^{\mathsf{AC}^0}$ reductions, but are also able to show that $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$ in this case.)

In either case, we obtain the consequence $\mathsf{NP} \neq \mathsf{MA} \cap \mathsf{P/poly}$. □

We close this section with another variant of Theorem 10.13, proved via the same technique:

**Theorem 10.17.** *Let* $\mathsf{NP}^A \subseteq \mathsf{P}^A/\mathsf{poly}$. *If* $\mathsf{NEXP} \not\subseteq \mathsf{PSPACE}^A$ *(or* $\mathsf{NEXP} \not\subseteq \mathsf{EXP}^A$*), then neither* $\mathrm{MKTP}^A$ *nor* $\mathrm{MCSP}^A$ *is hard for* $\mathsf{NP}$ *under logspace-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions (*$\mathsf{P}$*-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions, respectively).*

**Corollary 10.18.** $\mathrm{MKTP}^{\mathsf{QBF}}$ *is not hard for* $\mathsf{NP}$ *under logspace-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions (*$\mathsf{P}$*-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions) unless* $\mathsf{PSPACE} = \mathsf{NEXP}$ *(*$\mathsf{EXP} = \mathsf{NEXP}$*, respectively). The same holds for* $\mathrm{MCSP}^{\mathsf{QBF}}$.

Although the following corollary discusses $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reductions, it also says something about $\leq_{\mathrm{T}}^{\mathsf{L}}$ reducibility. This is because, assuming $\mathsf{DSPACE}(n) \not\subseteq \mathsf{i.o.SIZE}^{\mathsf{MKTP}^A}(2^{\epsilon n})$, any $\leq_{\mathrm{T}}^{\mathsf{L}}$ reduction to MKTP can be simulated by a logspace-uniform $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ reduction to MKTP. (To see this, note that, by Theorem 10.5, MKTP is hard for $\mathsf{DET}$ under this class of reductions, and hence each of the logspace-computable (nonadaptive) queries can be computed using oracle gates for MKTP, and similarly the logspace computation that uses the queries can also be simulated using MKTP. Similar observations arise in [AO96].)

**Corollary 10.19.** *If either of* MKTP *or* MCSP *is hard for* $\mathsf{NP}$ *under logspace-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions (*$\mathsf{P}$*-uniform* $\leq_{\mathrm{T}}^{\mathsf{AC}^0}$ *reductions), then* $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ *or* $\mathsf{NEXP} = \mathsf{PSPACE}$ *(*$\mathsf{NEXP} = \mathsf{EXP}$*, respectively).*

# Chapter 11

# Natural NP-Intermediate Problems

A problem in NP is called NP-*intermediate* if it is neither solvable in P nor NP-complete. It has been known since the work of Ladner [Lad75] that some NP-intermediate problem exists under the weakest assumption that $\mathsf{P} \neq \mathsf{NP}$. However, no "natural" NP-intermediate problem was known under similar complexity-theoretic assumptions. The NP-intermediate problem constructed by Ladner is very artificial because it is constructed by a diagonalization argument.

Problems such as factoring and Graph Isomorphism are sometimes put forward as candidates for NP-intermediate problems. Indeed, these problems are in $\mathsf{NP} \cap \mathsf{coAM}$ [Bab85], and hence these problems cannot be NP-complete unless $\mathsf{NP} \subseteq \mathsf{coAM}$. On the other hand, there is no strong complexity-theoretic argument explaining why these problems should not lie in P. (It should be noted that the recent breakthrough result of Babai [Bab16] showed a quasi-polynomial-time algorithm for Graph Isomorphism.)

In this chapter, we show the first natural NP-intermediate problems under very weak complexity-theoretic assumptions. We show that if $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ then approximating the size of the maximum clique in a graph within a factor of $n^{1-o(1)}$ is NP-intermediate. We also show that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is NP-intermediate for $\epsilon(n) = o(1)$ under the existence of auxiliary-input one-way functions.

## 11.1 GapMCSP is NP-Intermediate

In this section, we show that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is NP-intermediate when $\epsilon(n) = o(1)$ and an auxiliary-input one-way function exists. We first observe that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is equivalent to the following optimization problem. (See also [Gol06] for similar comments.)

**Fact 11.1.** $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *is polynomial-time Turing equivalent to the following approximation problem: Given a truth table $T$ of length $2^n$, the task is to output a value $f(T) \in \mathbb{N}$ such that*

$$\mathsf{size}(T) \leq f(T) \leq 2^{(1-\epsilon(|T|))\cdot n} \cdot \mathsf{size}(T).$$

Note that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ becomes easier when $\epsilon$ becomes smaller. If $\epsilon(n) = o(1)$, then it is easy to see that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ can be computed in $\mathsf{DTIME}(2^{n^{o(1)}})$. Here we show that a much faster algorithm can be obtained when MCSP reduces to $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ via a polynomial-time reduction.

**Theorem 11.2.** *For any efficiently computable nonincreasing $\epsilon(n) = o(1)$, if* $\mathrm{MCSP} \in \mathsf{P}^{\mathrm{Gap}_\epsilon\mathrm{MCSP}}$ *then* $\mathrm{MCSP} \in \mathsf{P}$.

The idea is that the $\mathrm{Gap}_\epsilon$MCSP is "strongly downward self-reducible." We will show that any $\mathrm{Gap}_\epsilon$MCSP instance of length $n$ is reducible to $n^{1-\epsilon}$ MCSP instances of length $n^\epsilon$. To this end, we will exploit the following simple fact.

**Lemma 11.3.** *For a function $f\colon \{0,1\}^n \to \{0,1\}$, a string $x \in \{0,1\}^k$ and $k \in \mathbb{N}$, let $f_x\colon \{0,1\}^{n-k} \to \{0,1\}$ be a function defined as $f_x(y) := f(x,y)$. Then, the following holds:*

$$\max_{x\in\{0,1\}^k} \mathrm{size}(f_x) \leq \mathrm{size}(f) \leq 2^k \cdot \left( \max_{x\in\{0,1\}^k} \mathrm{size}(f_x) + 3 \right),$$

*(In other words, $\max_{x\in\{0,1\}^k} \mathrm{size}(f_x)$ gives an approximation of $\mathrm{size}(f)$ within a factor of $2^k$.)*

*Proof.* We first claim that $\max_{x\in\{0,1\}^k} \mathrm{size}(f_x) \leq \mathrm{size}(f)$. Indeed, let $C$ be a minimum circuit that computes $f$ and let $x$ be an arbitrary string of length $k$. For each $x \in \{0,1\}^k$, define a circuit $C_x$ as $C_x(y) := C(x,y)$ on input $y \in \{0,1\}^{n-k}$. Then, since $C_x$ computes $f_x$ and the size of $C_x$ is at most that of $C$, we have $\mathrm{size}(f_x) \leq \mathrm{size}(f)$.

Next, we claim that $\mathrm{size}(f) \leq 2^k \cdot \left(\max_{x\in\{0,1\}^k} \mathrm{size}(f_x) + O(1)\right)$. For any $x \in \{0,1\}^k$, let $C_x$ be a minimum circuit that computes $f_x$. We build a circuit that computes $f =: f_\epsilon$ recursively as follows: $f_z(x,y) = (\neg x_1 \wedge f_{z0}(x_2,\ldots,x_k,y)) \vee (x_1 \wedge f_{z1}(x_2,\ldots,x_k,y))$ for any string $z$ of length less than $k$, and $f_x(y) = C_x(y)$ for any $x \in \{0,1\}^k$. Since $\mathrm{size}(f_z) \leq \mathrm{size}(f_{z0}) + \mathrm{size}(f_{z1}) + 3$ we obtain

$$\begin{aligned} \mathrm{size}(f) &\leq \sum_{x\in\{0,1\}^k} C_x(y) + 3\cdot(2^k - 1) \\ &< 2^k \cdot \left( \max_{x\in\{0,1\}^k} \mathrm{size}(f_x) + 3 \right). \end{aligned}$$

□

*Proof of Theorem 11.2.* Let $M$ be a polynomial-time oracle machine which reduces MCSP to $\mathrm{Gap}_\epsilon$MCSP. Let $|T|^c$ be an upper bound for the running time of $M$, given a truth table $T$, and let $|T| = 2^n$.

We recursively compute the circuit complexity of $T$ by the following procedure: Run $M$ on input $T$. If $M$ makes a query $S$ to the $\mathrm{Gap}_\epsilon$MCSP oracle, then divide $S$ into consecutive substrings $S_1, \cdots, S_{2^k}$ of length $|S| \cdot 2^{-k}$ such that $S_1 \cdot S_2 \cdots S_{2^k} = S$ (where $k$ is a parameter, chosen later, that depends on $|S|$), and compute the circuit complexity of each $S_i$ recursively for each $i \in [2^k]$. Then continue the simulation of $M$, using the value $2^k \cdot \left(\max_{i\in[2^k]} \mathrm{size}(S_i) + 3\right)$ as an approximation to $\mathrm{size}(S)$.

We claim that the procedure above gives the correct answer. It suffices to claim that the simulation of $M$ is correct in the sense that every query of $M$ is answered with a value that satisfies the approximation criteria of $\mathrm{Gap}_\epsilon$MCSP. Suppose that $M$ makes a query $S$. By the assumption on the running time of $M$, we have $|S| \leq |T|^c = 2^{nc}$. By Lemma 11.3, we have

$$\mathrm{size}(S) \leq 2^k \cdot \left( \max_{i\in[2^k]} \mathrm{size}(S_i) + 3 \right) \leq 2^k \cdot (\mathrm{size}(S) + 3).$$

In particular, the estimated value satisfies the promise of $\mathrm{Gap}_\epsilon$MCSP if $2^k \cdot (\mathrm{size}(S) + 3) \leq |S|^{1-\epsilon(|S|)} \cdot \mathrm{size}(S)$. Since we may assume without loss of generality

that $\mathrm{size}(S) \geq 3$, it suffices to make sure that $2^{k+1} \cdot \mathrm{size}(S) \leq |S|^{1-\epsilon(|S|)} \cdot \mathrm{size}(S)$. Let $|S| = 2^m$. Then, in order to satisfy $k+1 \leq (1-\epsilon(|S|)) \cdot m$, let us define $k := (1-\epsilon(|S|)) \cdot m - 1$. For this particular choice of $k$, the estimated value $2^k \cdot \left(\max_{i \in [2^k]} \mathrm{size}(S_i) + 3\right)$ of the circuit complexity of $S$ satisfies the promise of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, which implies that the reduction $M$ computes the correct answer for MCSP.

Now we analyze the time complexity of the algorithm. Each recursive step makes at most $2^{2cn}$ many recursive calls, because there are potentially $2^{cn}$ many queries $S$ of $M$, each of which may produce at most $2^k \leq 2^{cn}$ recursive calls. The length of each truth table $S_i$ that arises in one of the recursive calls is $|S_i| = |S| \cdot 2^{-k} = 2^{m-k} = 2^{\epsilon(|S|) \cdot m + 1}$. We claim that $|S_i| \leq 2^{1+(n/2)}$ holds for sufficiently large $n$. Let us take $n$ to be large enough so that $\epsilon(2^{n/2}) \leq 1/2c$. If $m \geq n/2$, then $|S_i| \leq 2^{\epsilon(2^m) \cdot m + 1} \leq 2^{\epsilon(2^{n/2}) \cdot cn + 1} \leq 2^{1+(n/2)}$. Otherwise, since $m \leq n/2$ and $\epsilon(|S|) < 1$, we obtain $|S_i| \leq 2^{\epsilon(|S|) \cdot m + 1} \leq 2^{1+(n/2)}$. Therefore, on inputs of length $2^n$, each recursive call produces instances of length at most $2^{1+(n/2)}$. The overall time complexity can be estimated as $2^{c'n} \cdot 2^{c'n/2} \cdot 2^{c'n/4} \cdots = 2^{2c'n}$ for some constant $c'$ (say, $c' = 3c$), which is a polynomial in the input length $2^n$. □

**Remark.** If we drop the assumption that $\epsilon(n)$ be computable, then the proof of Theorem 11.2 still shows that if $\mathrm{MCSP} \in \mathsf{P}^{\mathrm{Gap}_\epsilon\mathrm{MCSP}}/\mathsf{poly}$ then $\mathrm{MCSP} \in \mathsf{P}/\mathsf{poly}$.

**Corollary 11.4.** *Let $\epsilon(n) = o(1)$. If $\mathrm{Gap}_\epsilon\mathrm{MCSP} \notin \mathsf{P}/\mathsf{poly}$ then $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is not hard for $\mathsf{NP}$ (or even for MCSP) under $\leq_{\mathrm{T}}^{\mathsf{P}/\mathsf{poly}}$ reductions, and is thus $\mathsf{NP}$-intermediate.*

*Proof.* This is immediate from the preceding remark. If $\mathrm{MCSP} \in \mathsf{P}^{\mathrm{Gap}_\epsilon\mathrm{MCSP}}/\mathsf{poly}$ then $\mathrm{MCSP} \in \mathsf{P}/\mathsf{poly}$, which in turn implies that $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{P}/\mathsf{poly}$. □

In what follows, we show that the assumption of Corollary 11.4 is true under very modest cryptographic assumptions. Specifically, we assume the existence of auxiliary-input one-way functions secure against *polynomial-size circuits*. (We note that, in Chapter 3, the security of auxiliary-input one-way function is defined with respect to randomized polynomial-time machine; here we need a stronger notion of security.)

**Theorem 11.5.** *If an auxiliary-input one-way function secure against polynomial-size circuits exists, then there is a function $\epsilon(n) = o(1)$ such that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is $\mathsf{NP}$-intermediate. (Namely, $\mathrm{Gap}_\epsilon\mathrm{MCSP} \notin \mathsf{P}/\mathsf{poly}$ and $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ is not $\mathsf{NP}$-hard under $\leq_{\mathrm{T}}^{\mathsf{P}/\mathsf{poly}}$ reductions.)*

*Proof Sketch.* Let $f$ be an auxiliary-input one-way function secure against any polynomial-size circuit. Let $S(n)$ be the size of the smallest circuit $A$ such that for some $y \in \{0,1\}^n$ of length $n$, $\Pr_x[f_y(A(f_y(x))) = f_y(x)] \geq 1/2$ where $n = |y|$ and $x$ is chosen uniformly at random. By assumption, $S(n)$ is not bounded by any polynomial. Let $\epsilon(n)$ be a nondecreasing unbounded function such that $n^{c/\epsilon(n^c)} < S(n)$ for infinitely many $n$, where $c$ is a constant that we will pick later.

Observe that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ defines a natural property useful against $\mathsf{SIZE}(2^{\epsilon(n)n})$. By inspecting the reduction from an auxiliary-input one-way function to a natural property (Theorem 3.7), one can show that there exists an $n^{O(1/\epsilon(n^{O(1)}))}$-size oracle circuit that inverts the auxiliary-input one-way function $f$ with oracle access to $\mathrm{Gap}_\epsilon\mathrm{MCSP}$. If $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{P}/\mathsf{poly}$, then we obtain a circuit of size $n^{O(1/\epsilon(n^{O(1)}))}$ that inverts $f$. However, this is a contradiction for a large enough $c$. □

**Remark.** Observe that Theorem 11.5 can also be rephrased in terms of *uniform* probabilistic adversaries, if we assume that the one-way functions require time $n^{e(n)}$ to invert, for some easy-to-compute function $e$.

## 11.2 Reductions among GapMCSPs Require Large Stretch

In the previous section, we studied $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ where $\epsilon(n) = o(1)$. Our results do not rule out the possibility of $\mathsf{NP}$-hardness of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ when $\epsilon$ is a fixed positive constant. Here we observe that, even if $\epsilon$ is a constant, a similar proof technique enables us to show that a reduction requires large stretch: We say that a reduction from $\mathrm{Gap}_\delta\mathrm{MCSP}$ to $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *has stretch* $n^c$ if, on input $T$, the reduction makes queries of length at most $|T|^c$.

**Theorem 11.6.** *Let* $0 < \epsilon < \delta < 1$. *If* $\mathrm{Gap}_\delta\mathrm{MCSP}$ *is reducible to* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *via a polynomial-time Turing reduction of stretch at most* $n^c$ *for some* $c < \delta/\epsilon$, *then* $\mathrm{Gap}_\delta\mathrm{MCSP} \in \mathsf{P}$.

*Proof.* The argument is almost identical to the argument in the preceding section. Given an input to $\mathrm{Gap}_\delta\mathrm{MCSP}$, simulate the reduction from $\mathrm{Gap}_\delta\mathrm{MCSP}$ to $\mathrm{Gap}_\epsilon\mathrm{MCSP}$. As before, if the reduction makes a query $S$, then divide $S$ into consecutive substrings $S_1, \ldots, S_{2^k}$ of length $2^{m-k}$, where $m$ is defined as $|S| = 2^m$ and $k$ is a parameter chosen later depending on $m$. For each $i \in [2^k]$, recursively solve $\mathrm{Gap}_\delta\mathrm{MCSP}$ on the instance $S_i$, and let $f(S_i)$ be the answer of the recursive call. Now, we estimate the circuit complexity of $S$ as $2^k \cdot (\max_{i\in[2^k]} f(S_i) + 3)$ and continue the simulation.

We claim the correctness of the simulation for a certain choice of parameter $k = k(m)$. Let $e$ denote the estimated circuit complexity of $S$, that is, $e := 2^k \cdot (\max_{i\in[2^k]} f(S_i) + 3)$. The goal is to show that $e$ satisfies the promise of $\mathrm{Gap}_\epsilon\mathrm{MCSP}$, or equivalently,

$$\mathrm{size}(S) \le e \le |S|^{1-\epsilon} \cdot \mathrm{size}(S). \tag{11.1}$$

We may assume that answers of recursive calls satisfy the promise of $\mathrm{Gap}_\delta\mathrm{MCSP}$ by induction: that is, $\mathrm{size}(S_i) \le f(S_i) \le |S_i|^{1-\delta} \cdot \mathrm{size}(S_i)$. Thus, by Lemma 11.3, we have

$$e \ge 2^k \cdot \left( \max_{i\in[2^k]} \mathrm{size}(S_i) + 3 \right) \ge \mathrm{size}(S),$$

as required in the first inequality of (11.1). Now we turn to the second inequality of (11.1). We may assume, without loss of generality, that $e \le 2^{k+1} \cdot \max_{i\in[2^k]} f(S_i)$. Therefore, we obtain

$$\begin{aligned}
e &\le 2^{k+1} \cdot \max_{i\in[2^k]} f(S_i) \\
&\le 2^{k+1} \cdot \max_{i\in[2^k]} |S_i|^{1-\delta} \cdot \mathrm{size}(S_i) && \text{(by the promise of Gap}^\delta\text{MCSP)} \\
&= 2^{k+1+(m-k)(1-\delta)} \cdot \max_{i\in[2^k]} \mathrm{size}(S_i) && \text{(since } |S_i| = 2^{m-k}\text{)} \\
&\le 2^{k+1+(m-k)(1-\delta)} \cdot \mathrm{size}(S) && \text{(by Lemma 11.3)} \\
&\le |S|^{1-\epsilon} \cdot \mathrm{size}(S),
\end{aligned}$$

where the last inequality holds if $k + 1 + (m-k)(1-\delta) \le m \cdot (1-\epsilon)$, that is, $k \le m - m\epsilon/\delta - 1/\delta$. Thus we define $k$ as $k := m - m\epsilon/\delta - 1/\delta$, which ensures the second inequality of (11.1).

Now we turn to analysis of the running time of the algorithm. Let $2^n$ be the length of the input to the algorithm. By the assumption on the stretch of the reduction, we have $|S| \leq 2^{nc}$, that is, $m \leq nc$. Therefore, $|S_i| = 2^{m-k} = 2^{m\epsilon/\delta+1/\delta} \leq 2^{nc\epsilon/\delta+1/\delta}$. Since $c\epsilon/\delta < 1$, the algorithm above runs in polynomial time. (Indeed, let $t(N)$ be an upper bound of the running time of the algorithm on inputs of length $N$ and $\rho := c\epsilon/\delta < 1$. We have $t(N) \leq N^{O(1)}t(N^\rho)$. Solving this recursive inequality, we obtain $t(N) = N^{O(1)}$.) □

## 11.3 Approximating Maximum Clique is NP-Intermediate

We observe that the strongly downward self-reducibility property that we exploited above is fairly common. For instance, it has been noticed previously that CLIQUE also has this property [Sri03, AK10]. Thus a similar proof technique enables us to show NP-intermediateness of the following problem.

**Definition 11.7.** *For any function $\epsilon\colon \mathbb{N} \to (0,1)$, let* $\mathrm{Gap}_\epsilon$CLIQUE *be the approximation problem that, given an $n$-vertex graph $G$, asks for outputting a value $f(G) \in \mathbb{N}$ such that*

$$\omega(G) \leq f(G) \leq n^{1-\epsilon(n)} \cdot \omega(G).$$

*Here, as usual $\omega(G)$ denotes the clique number of $G$: the size of the largest clique in $G$.*

**Theorem 11.8.** NP $\not\subseteq$ P/poly *if and only if there is an $\epsilon(n) = o(1)$ such that* $\mathrm{Gap}_\epsilon$CLIQUE *has no solution in* P/poly *and is not hard for* NP *under* $\leq_{\mathrm{T}}^{\mathsf{P/poly}}$ *reductions.*

*Proof.* Assume NP $\not\subseteq$ P/poly. Define $e(n)$ to be the least $c$ such that, for all $m \leq n$, there is a circuit of size $m^c + c$ that computes a function $f(G)$ (for $m$-vertex graphs $G$) such that $\omega(G) \leq f(G) \leq m^{1-1/c} \cdot \omega(G)$. If $e(n) = O(1)$, it follows from [Hås99] that CLIQUE $\in$ P/poly, contrary to assumption. Thus $e(n) = \omega(1)$.

Let $\epsilon = \epsilon(n) = 1/e(n)$; thus $\epsilon(n) = o(1)$. It follows immediately from the definition of $e(n)$ that $\mathrm{Gap}_\epsilon$CLIQUE has no solution in P/poly.

If we partition the vertices of an $n$-node graph $G$ into $n^{1-\epsilon}$ parts $V_1, \ldots, V_{n^{1-\epsilon}}$ of size at most $\lceil n^\epsilon \rceil$, then $\omega(G) \leq (n^{1-\epsilon}) \cdot \max_i \omega(G_i)$, where $G_i$ is the induced subgraph of $G$ with vertices in $V_i$. (See [Sri03, AK10] for other applications of this observation.)

Now, precisely as in the proof of Theorem 11.2, it follows that if CLIQUE were P/poly-Turing reducible to $\mathrm{Gap}_\epsilon$CLIQUE, then CLIQUE $\in$ P/poly, contrary to our assumption. This shows that $\mathrm{Gap}_\epsilon$CLIQUE is not NP-hard under P/poly reductions, and thus completes the "only if" direction of the Theorem. (The converse is trivial.) □

# Chapter 12

# Unconditional Lower Bounds

It is widely believed that solving MCSP is hard; indeed, in Chapter 3, we showed that MCSP is not easy unless every one-way function is invertible. In this chapter, we present further evidence that MCSP is indeed a hard problem by establishing several unconditional circuit lower bounds for MCSP and its variants.

## 12.1 De Morgan Formula Lower Bounds for MCSP

In this section, we prove an unconditional formula lower bound for computing MCSP.

**Theorem 12.1.** *There exists a universal constant $d > 0$ such that* MCSP$[s]$ *requires a de Morgan formula of size $N^{2-o(1)}$ for any size parameter $s\colon \mathbb{N} \to \mathbb{N}$ such that $n^d \leq s(n) \leq n^{O(1)}$. Here $N := 2^n$ and $N$ denotes the length of inputs of* MCSP$[s]$.

A *De Morgan formula* is a circuit where the fan-out of each gate is at most 1 (i.e., the underlying graph is a tree), each internal gate is an AND or OR gate with fan-in 2, and the leaf is labelled with a literal (i.e., an input $x_i$ or its negation $\neg x_i$). We denote by $\mathrm{L}(\varphi)$ the size of a de Morgan formula $\varphi$, that is, the number of the literals that appear in $\varphi$.

The main idea of Theorem 12.1 is to use a *pseudorandom restriction*. Impagliazzo, Meka and Zuckerman [IMZ12] showed that the size of any formula hit with a pseudorandom restriction shrinks. We then observe that MCSP$[s]$ hit with a pseudorandom restriction does not become a trivial function unless the size parameter $s$ is too small, because each bit of a pseudorandom restriction can be efficiently computed.

We proceed to reviewing several ingredients. One of the fundamental results about de Morgan formulas is that a random restriction shrinks a formula. For a "restriction" $\rho\colon [N] \to \{0,1,*\}$ and a function $f\colon \{0,1\}^N \to \{0,1\}$, let $f\restriction_\rho$ denotes the function such that $f\restriction_\rho(x_1,\cdots,x_n) = f(y_1,\cdots,y_n)$ where $y_i := \rho(i)$ if $\rho(i) \in \{0,1\}$ and $y_i := x_i$ otherwise. In other words, $\rho(i) = *$ means that the variable is left unrestricted, and in the other case the variable is set to $\rho(i)$. We consider a *random restriction*, i.e., a distribution $\mathcal{R}$ of restrictions $\rho$. A random restriction $\rho \sim \mathcal{R}$ is said to be *$p$-regular* for $p > 0$ if $\Pr_{\rho\sim\mathcal{R}}[\rho(i) = *] = p$ for every $i \in [N]$ and $\Pr_{\rho\sim\mathcal{R}}[\rho(i) = b] = (1-p)/2$ for every $i \in [N]$ and every $b \in \{0,1\}$. We denote by $\mathcal{R}_p$ the distribution of $p$-regular random restrictions $\rho$ such that $\rho(i)$ is drawn from the $p$-regular distribution for each $i \in [N]$ *independently*.

**Lemma 12.2** (Håstad [Hås98], Tal [Tal14])**.** *Let $\varphi$ be any de Morgan formula.*

*Then, for every $p > 0$, it holds that*

$$\mathop{\mathbb{E}}_{\rho\sim\mathcal{R}_p}[\mathrm{L}(\varphi|_\rho)] = O\left(p^2\mathrm{L}(\varphi) + \sqrt{p^2\mathrm{L}(\varphi)}\right).$$

Lemma 12.2 shows that a formula shrinks quadratically under a random restriction $\rho \sim \mathcal{R}_p$. However, $\mathrm{MCSP}[s]|_\rho$ becomes a trivial function with high probability, since almost all the inputs are set to uniformly random bits. We thus need to use a *pseudorandom restriction*. Impagliazzo, Meka and Zuckerman [IMZ12] showed that there is a distribution $\mathcal{R}'$ such that for every restriction $\rho \in \mathsf{supp}(\mathcal{R}')$, the circuit complexity $\mathsf{size}(\rho)$ of $\rho$ is small. (Here we identify $\rho\colon [N] \to \{0,1,*\}$ with $\rho \in \{0,1,*\}^N$, and $\mathsf{size}(\rho) \le s$ means that there is a circuit $C$ of size at most $s$ such that $\rho(i) = C(i)$ for every $i \in [N]$.) We explain the idea below, following a simple exposition of [KRT17]. First, a De Morgan formula can be decomposed into small formulas in the following sense.

**Lemma 12.3** (Tal [Tal14])**.** *Let $\varphi$ be a de Morgan formula. Let $\ell \in \mathbb{N}$ be any parameter. Then there exist $m$ $(\le O(\mathrm{L}(\varphi)/\ell))$ formulas, denoted by $\varphi_1, \cdots, \varphi_m$, each of size at most $\ell$, and there exists a read-once formula $\psi$ of size $m$ such that $\psi(\varphi_1(x), \cdots, \varphi_m(x)) = \varphi(x)$ for every $x \in \{0,1\}^N$.*

In light of this, $\varphi$ shrinks even under $p$-regular $\ell$-wise independent random restrictions, since each small formula shrinks. Here we say that a random restriction $\rho$ is *$\ell$-wise independent* if every set $I$ $\left(\in \binom{[N]}{\ell}\right)$ of $\ell$ coordinates is independent, that is, $\Pr[\rho(i) = b_i \ (\forall i \in I)] = \prod_{i\in I}\Pr[\rho(i) = b_i]$ for every $b \in \{0,1,*\}^I$. There is a randomness-efficient way to sample such an $\ell$-wise independent random restrictions.

**Lemma 12.4** (Alon, Babai and Itai [ABI86])**.** *For every $N \in \mathbb{N}$, $p > 0$ and $\ell \in \mathbb{N}$, there exists a $p$-regular $\ell$-wise independent random restriction $\rho \sim \mathcal{R}_{p,\ell}$ such that $\mathsf{size}(\rho) \le \mathsf{poly}(\ell, \log N, \log\frac{1}{p})$ for every $\rho \in \mathsf{supp}(\mathcal{R}_{p,\ell})$. We denote this distribution by $\mathcal{R}_{p,\ell}$.*

**Lemma 12.5.** *There exists some universal constant $c > 1$ such that, for every de Morgan formula $\varphi$, for any parameter $p > 0$ and any $\ell = \Theta(p^{-2})$, it holds that*

$$\mathop{\mathbb{E}}_{\rho\sim\mathcal{R}_{p,\ell}}[\mathrm{L}(\varphi|_\rho)] \le cp^2\mathrm{L}(\varphi).$$

*Proof.* By Lemma 12.3, we decompose $\varphi$ so that $\psi(\varphi_1(x), \cdots, \varphi_m(x)) = \varphi(x)$ for every input $x$. Since $\psi$ is a read-once formula, we have $\mathrm{L}(\varphi|_\rho) \le \sum_{i=1}^m \mathrm{L}(\varphi_i|_\rho)$ for every restriction $\rho$. Therefore,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\rho\sim\mathcal{R}_{p,\ell}}[\mathrm{L}(\varphi|_\rho)] \\
&\le \sum_{i=1}^m \mathop{\mathbb{E}}_{\rho\sim\mathcal{R}_{p,\ell}}[\mathrm{L}(\varphi_i|_\rho)] \\
&= \sum_{i=1}^m \mathop{\mathbb{E}}_{\rho\sim\mathcal{R}_p}[\mathrm{L}(\varphi_i|_\rho)] && \text{(since } \varphi_i \text{ depends on } \le \ell \text{ variables)} \\
&\le \sum_{i=1}^m O\left(p^2\mathrm{L}(\varphi_i) + \sqrt{p^2\mathrm{L}(\varphi_i)}\right) && \text{(by Lemma 12.2)} \\
&\le O(m) = O(\mathrm{L}(\varphi)/\ell) = O(p^2\mathrm{L}(\varphi)).
\end{aligned}$$

□

In Lemma 12.5, we cannot take $p$ to be small enough for our purpose; however, by composing independent $p$-regular random restrictions $r$ times, we can reduce $p$ while keeping $\mathsf{size}(\rho)$ small. Here the *composition* $\rho_1\rho_2$ of two restrictions $\rho_1, \rho_2$ is defined naturally as follows: $\rho_1\rho_2(i) = \rho_1(i)$ if $\rho_1(i) \in \{0,1\}$ and $\rho_1\rho_2(i) = \rho_2(i)$ otherwise, for each $i \in [N]$. Let $\mathcal{R}^r_{p,\ell}$ denote the distribution of the composition of $r$ independent $p$-regular $\ell$-wise independent random restrictions. In particular, $\mathcal{R}^r_{p,\ell}$ is $p^r$-regular and $\ell$-wise independent.

**Lemma 12.6.** *There exists some universal constant $c > 1$ such that, for every de Morgan formula $\varphi$, for any parameters $p > 0, r \in \mathbb{N}$, $\ell = \Theta(p^{-2})$, it holds that*

$$\mathop{\mathbb{E}}_{\rho\sim\mathcal{R}^r_{p,\ell}}[\mathrm{L}(\varphi\restriction_\rho)] \le c^r p^{2r}\mathrm{L}(\varphi).$$

*Proof.* By induction on $r \ge 1$. Fix any restriction $\sigma \in \mathsf{supp}(\mathcal{R}^{r-1}_{p,\ell})$.

We pick a random restriction $\rho \sim \mathcal{R}_{p,\ell}$. By applying Lemma 12.6 for $\varphi\restriction_\sigma$, we obtain

$$\mathop{\mathbb{E}}_{\rho\sim\mathcal{R}_{p,\ell}}[\mathrm{L}((\varphi\restriction_\sigma)\restriction_\rho)] \le cp^2\mathrm{L}(\varphi\restriction_\sigma).$$

By averaging this inequality under the distribution $\sigma \sim \mathcal{R}^{r-1}_{p,\ell}$, we obtain

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma\rho\sim\mathcal{R}^r_{p,\ell}}[\mathrm{L}(\varphi\restriction_{\sigma\rho})] &\le cp^2 \mathop{\mathbb{E}}_{\sigma\sim\mathcal{R}^{r-1}_{p,\ell}}[\mathrm{L}(\varphi\restriction_\sigma)] \\
&\le cp^2 \cdot c^{r-1}p^{2(r-1)}\mathrm{L}(\varphi) \qquad \text{(by the induction hypothesis)} \\
&= c^r p^{2r}\mathrm{L}(\varphi).
\end{aligned}$$

□

Now we argue that, for a pseudorandom restriction $\rho \in \mathcal{R}^r_{p,\ell}$, $\mathrm{MCSP}[s]\restriction_\rho$ does not become a trivial function with high probability. Specifically, we show that $\mathrm{MCSP}[s]\restriction_\rho$ depends on almost all variables left unrestricted.

**Lemma 12.7.** *Assume $s \ge n$. Let $\rho\colon [N] \to \{0,1,*\}$ be a restriction such that $\mathsf{size}(\rho) \le s - O(1)$. Let $V$ be $\rho^{-1}(*)$, that is, the set of all the indices of variables left unrestricted by $\rho$. Then, for any formula $\varphi$ computing $\mathrm{MCSP}[s]$, we have $\mathrm{L}(\varphi\restriction_\rho) \ge |V| - O(s \log s)$.*

*Proof.* Let $V_0 \subseteq V$ be the set of variables on which $\varphi\restriction_\rho$ does not depend. It suffices to claim that $|V_0| = O(s \log s)$ because $\mathrm{L}(\varphi\restriction_\rho) \ge |V| - |V_0|$.

Let $\bar{0}\colon [N] \to \{0,1,*\}$ denote the constant-0 function. Consider an input $\rho\bar{0} \in \{0,1\}^N$, that is, the string where each $*$ of $\rho$ is replaced with 0. Since $\mathsf{size}(\rho) \le s - O(1)$, one can easily observe that $\mathsf{size}(\rho\bar{0}) \le s$. Therefore, $\rho\bar{0}$ is an YES instance of $\mathrm{MCSP}[s]$. Since $\varphi\restriction_\rho$ does not depend on $V_0$, for every assignment $\sigma\colon V_0 \to \{0,1\}$, $\varphi$ also accepts the input $\rho\sigma\bar{0}$. Now by counting the number of YES instances in $\mathrm{MCSP}[s]$, we obtain

$$\begin{aligned}
2^{|V_0|} &\le \#\{\, x \in \{0,1\}^N \mid \ \varphi \text{ accepts } x \ \} \\
&= \#\{\, x \in \{0,1\}^N \mid x \in \mathrm{MCSP}[s] \,\} \le 2^{O(s\log s)}.
\end{aligned}$$

□

We are now ready to prove unconditional de Morgan formula lower bounds.

*Proof of Theorem 12.1.* Let $\varphi$ be any de Morgan formula computing MCSP[$s$]. Fix any small constant $\epsilon > 0$.

By Lemma 12.6, for any parameters $p > 0, r \in \mathbb{N}$, $\ell = \Theta(p^{-2})$, it holds that

$$\mathop{\mathbb{E}}_{\rho \sim \mathcal{R}^r_{p,\ell}} [\mathrm{L}(\varphi\restriction_\rho)] \leq c^r p^{2r} \mathrm{L}(\varphi). \tag{12.1}$$

Observe that $c^r p^{2r} = (p^r)^{2-\log_{1/p} c} = (p^r)^{2-\epsilon}$, where we fix $p$ to be a small enough constant so that $\epsilon = \log_{1/p} c$.

On the other hand, by Lemma 12.7,

$$\mathrm{L}(\varphi\restriction_\rho) \geq |\rho^{-1}(*)| - O(s \log s), \tag{12.2}$$

for every $\rho$ such that $\mathsf{size}(\rho) \leq s - O(1)$.

We will take $r$ so that $r \leq \log N = n$. By the construction of $\mathcal{R}^r_{p,\ell}$, we have $\mathsf{size}(\rho) \leq \mathsf{poly}(r, \ell, \log N) = \mathsf{poly}(1/p^2, n) = \mathsf{poly}(n)$ for every $\rho \in \mathsf{supp}(\mathcal{R}^r_{p,\ell})$. Thus there exists a universal constant $d$ such that $\mathsf{size}(\rho) \leq n^d - O(1)$ for all large $n \in \mathbb{N}$. We assume that $n^d \leq s$ as the hypothesis of Theorem 12.1.

Therefore, we can combine (12.1) and (12.2) to obtain

$$\begin{aligned}(p^r)^{2-\epsilon} \mathrm{L}(\varphi) &\geq \mathop{\mathbb{E}}_{\rho \sim \mathcal{R}^r_{p,\ell}} [|\rho^{-1}(*)| - O(s \log s)] = p^r N - O(s \log s) \\ &\geq p^r N/2,\end{aligned}$$

where, in the last inequality, we take $r$ so that $O(s \log s) \leq p^r N/2 \leq n^{O(1)}$. We thus obtain

$$\mathrm{L}(\varphi) \geq \frac{N}{(p^r)^{1-\epsilon}} \geq \frac{N}{(n^{O(1)}/N)^{1-\epsilon}} = N^{2-\epsilon}/n^{O(1)} \geq N^{2-2\epsilon}$$

for all large $n \in \mathbb{N}$. □

An obvious open question is to prove a better formula lower bound.

**Open Question 12.8.** Show that MCSP requires a de Morgan formula of size $\omega(N^2)$.

## 12.2 Average-case $\mathsf{AC}^0[p]$ Lower Bound of MKTP

In this section, we show an unconditional average-case $\mathsf{AC}^0[p]$ circuit lower bound of MKTP. The whole section is devoted to proving the following result:

**Theorem 12.9.** *There exists some function $s(n)$ such that* (MKTP[$s$],$\mathcal{U}$) *is not in* $\mathsf{Avg}_\epsilon\mathsf{AC}^0[p]$ *for any prime $p$ and any constant $\epsilon \in (0,1)$.*

Our proof is based on the techniques of Fefferman, Shaltiel, Umans and Viola [FSUV13] They constructed a pseudorandom generator $G$ secure against $\mathsf{AC}^0[p]$ such that each output bit of $G$ has small KT-complexity. We first focus on the case when $p \neq 2$. In this case, we use the following pseudorandom generator $G$ based on $\mathsf{PARITY}$.

**Definition 12.10** ([FSUV13])**.** *For a parameter $k = k(m)$ chosen later, define $G\colon (\{0,1\}^m)^k \to \{0,1\}^{mk+k}$ as*

$$G(x_1, \ldots, x_k) := x_1 \cdots x_k \cdot \mathsf{PARITY}(x_1) \cdots \mathsf{PARITY}(x_k)$$

*for every $(x_1, \ldots, x_k) \in (\{0,1\}^m)^k$. Let $n := mk + k$.*

**Lemma 12.11** (implicit in [FSUV13])**.** *If there is an oracle $A$ that distinguishes $G$ from the uniform distribution with advantage $\epsilon$ for some constant $\epsilon > 0$, then there is an $\mathsf{AC}^0$ circuit $C$ with $A$-oracle gates such that $C^A(x) = \mathsf{PARITY}(x)$ for any input $x \in \{0,1\}^m$.*

*Proof Sketch.* For completeness, we include a brief proof sketch. They showed that, by using *resamplability* of $\mathsf{PARITY}$, there is an $\mathsf{NC}^0$ circuit $C_0$ with one $A$-oracle gate such that $\Pr_{x\sim\mathcal{U}_m}[C_0^A(x) = \mathsf{PARITY}(x)] \geq \frac{1+\epsilon}{2}$ ([FSUV13, Lemma 4.5]). By using resamplability again for $t$ independent choices of randomness (for some appropriately chosen $t$), we obtain circuits $C_1^A, \ldots, C_t^A$ each of which approximates $\mathsf{PARITY}$. Now taking the majority vote of these circuits, we can compute $\mathsf{PARITY}$ on all inputs. Here, the majority can be implemented by using Approximate Majority [AB84] in $\mathsf{AC}^0$, because the advantage of approximating $\mathsf{PARITY}$ is at least a constant $\epsilon$. As a result, we obtain an $\mathsf{AC}^0$ circuit with $A$-oracle gates that computes $\mathsf{PARITY}$ on all inputs ([FSUV13, Proposition 4.21]). □

Therefore, it is sufficient to claim that an errorless heuristic algorithm for MKTP[$s$] distinguishes the pseudorandom generator $G$. We first claim that the KT-complexity of any output of the pseudorandom generator $G$ in Lemma 12.11 is small.

**Claim 12.12.**

$$\mathrm{KT}(G(x_1, \ldots, x_k)) \leq mk + \mathsf{poly}(m, \log k)$$

*for every seed $(x_1, \ldots, x_k) \in (\{0,1\}^m)^k$.*

*Proof.* We use a description $d := (x_1, \ldots, x_k)$. Given an index $i \in \{1, \ldots mk + k\}$ of $G(x_1, \ldots, x_k)$, if $i \leq mk$ then output the $i$th bit of the description $d$; if $i > mk$ then compute and output $\mathsf{PARITY}(x_{i-mk})$, which takes $\mathsf{poly}(m, \log k)$ steps. □

Therefore, for a sufficiently large polynomial $k(m) = m^{O(1)}$, it holds that $\mathrm{KT}(G(x_1, \ldots, x_k)) \leq mk + o(k) \leq n - \log n =: s(n)$ (and thus an MKTP[$s$] oracle distinguishes $G$ from the uniform distribution).

Now assume, towards a contradiction, that there is an errorless heuristic $\mathsf{AC}^0[p]$ circuit $A_0$ that computes MKTP[$s$] with failure probability $\epsilon$. We define another circuit $A$ as $A(y) := 1$ if $A_0(y) = 1$ or $\perp$; otherwise $A(y) := 0$. Note that $A$ does not err on YES instances of MKTP[$s$]; that is, $A(G(x_1, \ldots, x_k)) = 1$ for every seed $(x_1, \ldots, x_k)$. On the other hand, consider the uniform distribution $y \sim \mathcal{U}_n$. Observe that the probability that $A_0(y) = \perp$ is at most $\epsilon$, and the probability that $y$ is an YES instance of MKTP[$s$] is at most $2^{-\log n+1} = o(1)$. Therefore, $\Pr_{y\sim\mathcal{U}_n}[A(y) = 0] \geq 1 - \epsilon - o(1)$. Hence $A$ distinguishes the output of $G$ from the uniform distribution with advantage $1 - \epsilon - o(1)$. Now we apply Lemma 12.11 to obtain an $\mathsf{AC}^0$ circuit $C^A$ with $A$-oracle gates that solves $\mathsf{PARITY}$. Since $A \in \mathsf{AC}^0[p]$, it shows that $\mathsf{PARITY} \in \mathsf{AC}^0[p]$, which contradicts the lower bounds of Razborov-Smolensky [Raz87, Smo87] for an odd prime $p$.

When $p = 2$, we use a pseudorandom generator $G_{\mathsf{CMD}}$ based on a problem called $\mathsf{CMD}$ (connectivity matrix determinant), which was introduced by Ishai and Kushilevitz [IK00, IK02]. For the exact definition of $G_{\mathsf{CMD}}$, the reader is referred to [FSUV13]. Here we only need the following property, which easily follows from the fact that $\mathsf{CMD}$ is computable in polynomial time.

**Fact 12.13.**

$$\mathrm{KT}(G_{\mathsf{CMD}}(x_1, \ldots, x_k)) \leq mk + \mathsf{poly}(m, \log k)$$

*for every seed $(x_1, \ldots, x_k) \in (\{0,1\}^m)^k$.*

**Lemma 12.14** ([FSUV13])**.** *If there is an oracle $A$ that distinguishes $G_{\mathsf{CMD}}$ from the uniform distribution with advantage a constant $\epsilon > 0$, then there is an $\mathsf{AC}^0[2]$ circuit $C$ with $A$-oracle gates such that $C^A(x) = \mathsf{MAJORITY}(x)$ for any $x \in \{0,1\}^m$.*

*Proof Sketch.* The problem $\mathsf{CMD}$ is resamplable in $\mathsf{AC}^0[2]$ ([FSUV13]), and hence as in Lemma 12.11, $\mathsf{CMD}$ can be solved by an $\mathsf{AC}^0[2]$ circuit with $A$-oracle gates. Since $\mathsf{CMD}$ is $\oplus\mathsf{L}$-complete under $\mathsf{NC}^0$ reductions ([IK02]), $\mathsf{MAJORITY}$ can be also solved by an $\mathsf{AC}^0[2]$ circuit with $A$-oracle gates. □

Combining Fact 12.13 and Lemma 12.14, we obtain an $\mathsf{AC}^0[2]$ circuit that solves $\mathsf{MAJORITY}$, which contradicts the lower bound of [Raz87, Smo87] for the majority function. This completes the proof of Theorem 12.9.

# Chapter 13

# Conclusions

In this thesis, we investigated the complexity of MCSP and its variants. Our results further highlight that improved understanding of MCSP will have major impacts in complexity theory. In particular, any hardness of $\mathrm{Gap}_\epsilon$MCSP implies further evidence that $\mathsf{DistNP} \not\subseteq \mathsf{AvgP}$, i.e., there is no efficient errorless heuristic algorithm solving $\mathsf{NP}$ problems on average (cf. Chapter 4). This can be seen as a significant step towards excluding Heuristica. We strongly believe that our results and techniques developed in this thesis are a key to understanding Heuristica well and advancing complexity theory further.

Currently, $\mathsf{NP}$-hardness of $\mathfrak{C}$-MCSP is known only for $\mathfrak{C} = \mathsf{DNF}$ or $\mathsf{DNF} \circ \mathsf{XOR}$ (cf. Chapter 9). Extending it to larger circuit classes $\mathfrak{C}$ is challenging as argued in Chapter 7. However, there is no fundamental barrier that prevents us from obtaining such a result. Thus we believe that it is possible to push this direction further, and moreover we believe that pushing this direction would give us new insight about a circuit class $\mathfrak{C}$.

Another potential approach for showing $\mathsf{NP}$-hardness of MCSP is via non-relativizing proof techniques. While there is some relativization barrier (cf. Chapter 4), a non-relativizing proof technique enables us to prove $\mathsf{NP}$-hardness of MCSP under the unlikely assumption that $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ [ABK+06b, IKV18]. Extending this proof technique to obtain $\mathsf{NP}$-hardness of MCSP unconditionally is left as an important open question, as it will exclude Heuristica. It should be noted that a similar situation happens often in complexity theory: Santhanam [San09] proved that $\mathsf{MA/1} \not\subseteq \mathsf{SIZE}(n^k)$ for every constant $k$ by analyzing two cases depending on whether $\mathsf{PSPACE} \subseteq \mathsf{P/poly}$ or not. Similarly, one might be able to prove $\mathsf{NP}$-hardness of MCSP assuming $\mathsf{PSPACE} \not\subseteq \mathsf{P/poly}$.

An important direction left as future work is to go beyond Heuristica: Can one construct a one-way function from hardness of MCSP? In the seminal work of Ajtai [Ajt96] about worst-case to average-case reductions, a one-way function was constructed based on the worst-case hardness of the shortest vector problem. As shown in Chapter 3, the existence of auxiliary-input one-way functions implies intractability of MCSP. As an intermediate step towards the existence of one-way functions, we pose the question of the converse direction: Can one construct auxiliary-input one-way functions from hardness of MCSP? For example:

**Open Question 13.1.** Show that an auxiliary-input one-way function exists if $\mathrm{Gap}_\epsilon\mathrm{MCSP} \notin \mathsf{P/poly}$ for every constant $\epsilon > 0$.

Subsequent to our work, the "hardness magnification" phenomenon was found by Oliveira and Santhanam [OS18], later with Pich [OPS18]. Very roughly speaking, given a circuit lower bound for MCSP$[s]$, one can construct another function of $s$-bit inputs that have the same circuit lower bound. Thus in particular, a very

weak circuit lower bound for MCSP[$s$] is enough to prove a circuit lower bound such as $\mathsf{NP} \not\subseteq \mathsf{P/poly}$. Unfortunately, the results presented in Chapter 12 are not enough to be combined with their results. First, one needs to compute an error-correcting code within a circuit class $\mathfrak{C}$ in order to apply the hardness magnification phenomenon. Thus the lower bound of MCSP[$s$] $\notin \mathsf{Formula}(N^{2-o(1)})$ cannot be combined with their ideas. Second, the size parameter $s$ must be small enough (e.g., $s(n) = (\log(n))^c$ for some fixed constant $c$, where $n$ denotes an input length). In our lower bound of MKTP[$s$] $\notin \mathsf{AC}^0[p]$, the size parameter $s(n)$ must be close to $n$, and thus the hardness magnification does not amplify the circuit lower bound.

It was an open question to extend our circuit lower bound MKTP $\notin \mathsf{AC}^0[p]$ to the case of MCSP. Very recently, the circuit lower bound MCSP $\notin \mathsf{AC}^0[p]$ was proved by Golovnev, Impagliazzo, Kabanets, Kolokolova, and Tal (personal communication). However, all the proof techniques for showing $\mathsf{AC}^0[p]$ circuit lower bounds do not seem to yield any lower bound for $\mathrm{Gap}_\epsilon$MCSP. An interesting open question is to prove $\mathsf{AC}^0[p]$ circuit lower bounds for a 2-factor approximation of MCSP or MKTP.

**Open Question 13.2.** Show that there is no $\mathsf{AC}^0[p]$ circuit that can approximate the minimum circuit size of a given truth table within a factor of 2.

More broadly, we believe that understanding the complexity of MCSP is not only a fundamental question itself, but also an approach of advancing complexity theory. We conjecture that there are still a number of exciting results about MCSP waiting to be found.

# References


[AAR98] Manindra Agrawal, Eric Allender, and Steven Rudich. Reductions in Circuit Complexity: An Isomorphism Theorem and a Gap Theorem. *J. Comput. Syst. Sci.*, 57(2):127–143, 1998.

[AB84] Miklós Ajtai and Michael Ben-Or. A Theorem on Probabilistic Constant Depth Computations. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 471–474, 1984.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[ABF+08] Michael Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1):16–34, 2008.

[ABFL14] Eric Allender, Harry Buhrman, Luke Friedman, and Bruno Loff. Reductions to the set of random strings: The resource-bounded case. *Logical Methods in Computer Science*, 10(3), 2014.

[ABG+14] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in $\mathsf{AC}^0 \circ \mathsf{MOD}_2$. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 251–260, 2014.

[ABI86] Noga Alon, László Babai, and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *J. Algorithms*, 7(4):567–583, 1986.

[ABK06a] Eric Allender, Harry Buhrman, and Michal Koucký. What can be efficiently reduced to the Kolmogorov-random strings? *Ann. Pure Appl. Logic*, 138(1-3):2–19, 2006.

[ABK+06b] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from Random Strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[ABX08] Benny Applebaum, Boaz Barak, and David Xiao. On Basing Lower-Bounds for Learning on Worst-Case Assumptions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 211–220, 2008.

[AD97] Miklós Ajtai and Cynthia Dwork. A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 284–293, 1997.

[AD17] Eric Allender and Bireswar Das. Zero knowledge and circuit minimization. *Inf. Comput.*, 256:2–8, 2017.

[ADF+13] Eric Allender, George Davie, Luke Friedman, Samuel Hopkins, and Iddo Tzameret. Kolmogorov Complexity, Circuits, and the Strength of Formal Theories of Arithmetic. *Chicago J. Theor. Comput. Sci.*, 2013, 2013.

[Adl78] Leonard M. Adleman. Two Theorems on Random Polynomial Time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 75–83, 1978.

[AFG13] Eric Allender, Luke Friedman, and William I. Gasarch. Limits on the computational power of random strings. *Inf. Comput.*, 222:80–92, 2013.

[AGGM06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 701–710, 2006.

[AGGM10] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. Erratum for: on basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 795–796, 2010.

[Agr11] Manindra Agrawal. The isomorphism conjecture for constant depth reductions. *J. Comput. Syst. Sci.*, 77(1):3–13, 2011.

[AGvM+18] Eric Allender, Joshua Grochow, Dieter van Melkebeek, Andrew Morgan, and Cristopher Moore. Minimum Circuit Size, Graph Isomorphism and Related Problems. *SIAM Journal on Computing*, 47:1339–1372, 2018.

[AH91] William Aiello and Johan Håstad. Statistical Zero-Knowledge Languages can be Recognized in Two Rounds. *J. Comput. Syst. Sci.*, 42(3):327–345, 1991.

[AH17] Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 54:1–54:14, 2017.

[AHK17] Eric Allender, Dhiraj Holden, and Valentine Kabanets. The Minimum Oracle Circuit Size Problem. *Computational Complexity*, 26(2):469–496, 2017.

[AHM+08] Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing Disjunctive Normal Form Formulas and $AC^0$ Circuits Given a Truth Table. *SIAM J. Comput.*, 38(1):63–84, 2008.

[Ajt96] Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 99–108, 1996.

[AK10] Eric Allender and Michal Koucký. Amplifying lower bounds by means of self-reducibility. *J. ACM*, 57(3):14:1–14:36, 2010.

[AKR+01] Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. Time-Space Tradeoffs in the Counting Hierarchy. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 295–302, 2001.

[AKRR11] 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.

[AKS98] Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. *Random Struct. Algorithms*, 13(3-4):457–466, 1998.

[Ale11] Michael Alekhnovich. More on Average Case vs Approximation Complexity. *Computational Complexity*, 20(4):755–786, 2011.

[All99] Eric Allender. The Permanent Requires Large Uniform Threshold Circuits. *Chicago J. Theor. Comput. Sci.*, 1999, 1999.

[All01] Eric Allender. When Worlds Collide: Derandomization, Lower Bounds, and Kolmogorov Complexity. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, 21st Conference, Bangalore, India, December 13-15, 2001, Proceedings*, pages 1–15, 2001.

[All12] Eric Allender. Curiouser and Curiouser: The Link between Incompressibility and Complexity. In *How the World Computes - Turing Centenary Conference and 8th Conference on Computability in Europe, CiE 2012, Cambridge, UK, June 18-23, 2012. Proceedings*, pages 11–16, 2012.

[All17] Eric Allender. The Complexity of Complexity. In *Computability and Complexity - Essays Dedicated to Rodney G. Downey on the Occasion of His 60th Birthday*, pages 79–94, 2017.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998.

[AMN98] Yossi Azar, Rajeev Motwani, and Joseph Naor. Approximating Probability Distributions Using Small Sample Spaces. *Combinatorica*, 18(2):151–171, 1998.

[AO96] Eric Allender and Mitsunori Ogihara. Relationships Among PL, #L, and the Determinant. *ITA*, 30(1):1–21, 1996.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP ∩ coNP. *J. ACM*, 52(5):749–765, 2005.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A New Barrier in Complexity Theory. *TOCT*, 1(1):2:1–2:54, 2009.

[Bab85] László Babai. Trading Group Theory for Randomness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 421–429, 1985.

[Bab16] László Babai. Graph isomorphism in quasipolynomial time [extended abstract]. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 684–697, 2016.

[BB15] Andrej Bogdanov and Christina Brzuska. On Basing Size-Verifiable One-Way Functions on NP-Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 1–6, 2015.

[BCGL92] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby. On the Theory of Average Case Complexity. *J. Comput. Syst. Sci.*, 44(2):193–219, 1992.

[BFKL10] Harry Buhrman, Lance Fortnow, Michal Koucký, and Bruno Loff. Derandomizing from Random Strings. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 58–63, 2010.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity*, 3:307–318, 1993.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP Question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BGSV16] Peter Bürgisser, Oded Goldreich, Madhu Sudan, and Salil Vadhan. Complexity Theory. *Oberwolfach Reports*, 12(4):3049–3099, 2016.

[BHK+16] Boaz Barak, Samuel B. Hopkins, Jonathan A. Kelner, Pravesh Kothari, Ankur Moitra, and Aaron Potechin. A Nearly Tight Sum-of-Squares Lower Bound for the Planted Clique Problem. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 428–437, 2016.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP Have Short Interactive Proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On Uniformity within $NC^1$. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.

[BM88] László Babai and Shlomo Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

[BM97] Harry Buhrman and Elvira Mayordomo. An Excursion to the Kolmogorov Random Strings. *J. Comput. Syst. Sci.*, 54(3):393–399, 1997.

[BM99] Andrej Brodnik and J. Ian Munro. Membership in Constant Time and Almost-Minimum Space. *SIAM J. Comput.*, 28(5):1627–1640, 1999.

[BR17] Andrej Bogdanov and Alon Rosen. Pseudorandom Functions: Three Decades Later. In *Tutorials on the Foundations of Cryptography.*, pages 79–158. 2017.

[BT06a] Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

[BT06b] Andrej Bogdanov and Luca Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[BU11] David Buchfuhrer and Christopher Umans. The complexity of Boolean formula minimization. *J. Comput. Syst. Sci.*, 77(1):142–153, 2011.

[Cai07] Jin-yi Cai. $S_2^P$ is subset of $ZPP^{NP}$. *J. Comput. Syst. Sci.*, 73(1):25–35, 2007.

[Can96] Ran Canetti. More on BPP and the Polynomial-Time Hierarchy. *Inf. Process. Lett.*, 57(5):237–241, 1996.

[CCHO05] Jin-yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005.

[CDE+14] Mingzhong Cai, Rodney G. Downey, Rachel Epstein, Steffen Lempp, and Joseph S. Miller. Random strings and tt-degrees of Turing complete C.E. sets. *Logical Methods in Computer Science*, 10(3), 2014.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 10:1–10:24, 2016.

[CIKK17] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Agnostic Learning from Tolerant Natural Proofs. In *Proceedings of the Approximation, Randomization, and Combinatorial Optimization (APPROX)*, pages 35:1–35:19, 2017.

[Coo71] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[CS16] Gil Cohen and Igor Shinkar. The Complexity of DNF of Parities. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 47–58, 2016.

[Czo99] Sebastian Czort. The Complexity of Minimizing Disjunctive Normal Form Formulas. Master's Thesis, University of Aarhus, 1999.

[DETT09] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved Pseudorandom Generators for Depth 2 Circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:141, 2009.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6):644–654, 1976.

[Fei98] Uriel Feige. A Threshold of ln $n$ for Approximating Set Cover. *J. ACM*, 45(4):634–652, 1998.

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 534–543, 2002.

[Fel09] Vitaly Feldman. Hardness of approximate two-level logic minimization and PAC learning with membership queries. *J. Comput. Syst. Sci.*, 75(1):13–26, 2009.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[FK00] Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, 16(2):195–208, 2000.

[FKO06] Uriel Feige, Jeong Han Kim, and Eran Ofek. Witnesses for non-satisfiability of dense random 3CNF formulas. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 497–508, 2006.

[FO07] Uriel Feige and Eran Ofek. Easily refutable subformulas of large random 3CNF formulas. *Theory of Computing*, 3(1):25–43, 2007.

[For89] Lance Fortnow. The Complexity of Perfect Zero-Knowledge. *Advances in Computing Research*, 5:327–343, 1989.

[FSUV13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On Beating the Hybrid Argument. *Theory of Computing*, 9:809–843, 2013.

[GG00] Oded Goldreich and Shafi Goldwasser. On the Limits of Nonapproximability of Lattice Problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GKM15] Parikshit Gopalan, Daniel M. Kane, and Raghu Meka. Pseudorandomness via the Discrete Fourier Transform. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 903–922, 2015.

[God] Chris Godsil. Double orthogonal complement of a finite module. MathOverflow (Available at `https://mathoverflow.net/q/75268`, Retrieved 19-01-2018).

[Gol06] Oded Goldreich. On Promise Problems: A Survey. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, pages 254–290, 2006.

[Gro98] Vince Grolmusz. A Lower Bound for Depth-3 Circuits with MOD $m$ Gates. *Inf. Process. Lett.*, 67(2):87–90, 1998.

[GS86] Shafi Goldwasser and Michael Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 59–68, 1986.

[GST07] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP Languages are Hard on the Worst-Case, Then it is Easy to Find Their Hard Instances. *Computational Complexity*, 16(4):412–441, 2007.

[GV08] Dan Gutfreund and Salil P. Vadhan. Limitations of Hardness vs. Randomness under Uniform Reductions. In *Proceedings of the Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX)*, pages 469–482, 2008.

[Hag91] Torben Hagerup. Fast Parallel Generation of Random Permutations. In *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, pages 405–416, 1991.

[Hås86] Johan Håstad. Almost Optimal Lower Bounds for Small Depth Circuits. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 6–20, 1986.

[Hås98] Johan Håstad. The Shrinkage Exponent of de Morgan Formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

[Hås99] Johan Håstad. Clique is hard to approximate withinn1 — $\epsilon$. *Acta Mathematica*, 182(1):105–142, March 1999.

[HILL99] 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.

[Hir15] Shuichi Hirahara. Identifying an Honest EXPˆNP Oracle Among Many. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 244–263, 2015.

[Hir18] Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.

[HK18] Shuichi Hirahara and Akitoshi Kawamura. On characterizations of randomized computation using plain Kolmogorov complexity. *Computability*, 7(1):45–56, 2018.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[HOS18] Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of Minimum Circuit Size Problem for OR-AND-MOD Circuits. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 5:1–5:31, 2018.

[HP15] John M. Hitchcock and Aduri Pavan. On the NP-Completeness of the Minimum Circuit Size Problem. In *Proceedings of the Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 236–245, 2015.

[HR12] Ishay Haviv and Oded Regev. Tensor-based Hardness of the Shortest Vector Problem to within Almost Polynomial Factors. *Theory of Computing*, 8(1):513–531, 2012.

[HS65] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HS17] Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 7:1–7:20, 2017.

[HW16] Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 18:1–18:20, 2016.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 294–304, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002.

[IKV18] Russell Impagliazzo, Valentine Kabanets, and Ilya Volkovich. The Power of Natural Properties as Oracles. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 7:1–7:20, 2018.

[IKW02] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.*, 65(4):672–694, 2002.

[IL89] Russell Impagliazzo and Michael Luby. One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 230–235, 1989.

[IL90] Russell Impagliazzo and Leonid A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 812–821, 1990.

[Imp95] Russell Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 134–147, 1995.

[Imp11] Russell Impagliazzo. Relativized Separations of Worst-Case and Average-Case Complexities for NP. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 104–114, 2011.

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from Shrinkage. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 111–119, 2012.

[IN96] Russell Impagliazzo and Moni Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *J. Cryptology*, 9(4):199–216, 1996.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if $E$ Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 220–229, 1997.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001.

[Jer92] Mark Jerrum. Large Cliques Elude the Metropolis Process. *Random Struct. Algorithms*, 3(4):347–360, 1992.

[JP00] Ari Juels and Marcus Peinado. Hiding Cliques for Cryptographic Security. *Des. Codes Cryptography*, 20(3):269–280, 2000.

[Juk06] Stasys Jukna. On Graph Complexity. *Combinatorics, Probability & Computing*, 15(6):855–876, 2006.

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[KC00] Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

[KLS+00] Marcos A. Kiwi, Carsten Lund, Daniel A. Spielman, Alexander Russell, and Ravi Sundaram. Alternation in interaction. *Computational Complexity*, 9(3-4):202–246, 2000.

[Ko91] Ker-I Ko. On the Complexity of Learning Minimum Time-Bounded Turing Machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[KRT17] Ilan Komargodski, Ran Raz, and Avishay Tal. Improved Average-Case Lower Bounds for De Morgan Formula Size: Matching Worst-Case Lower Bound. *SIAM J. Comput.*, 46(1):37–57, 2017.

[KS08] Subhash Khot and Rishi Saket. Hardness of Minimizing and Learning DNF Expressions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 231–240, 2008.

[Kuc95] Ludek Kucera. Expected Complexity of Graph Partitioning Problems. *Discrete Applied Mathematics*, 57(2-3):193–212, 1995.

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph Nonisomorphism Has Subexponential Size Proofs Unless the Polynomial-Time Hierarchy Collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[Lad75] Richard E. Ladner. On the Structure of Polynomial Time Reducibility. *J. ACM*, 22(1):155–171, 1975.

[Lev73] Leonid Anatolevich Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

[Lev84] Leonid A. Levin. Randomness Conservation Inequalities; Information and Independence in Mathematical Theories. *Information and Control*, 61(1):15–37, 1984.

[Lev86] Leonid A. Levin. Average Case Complete Problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[LL76] Richard E. Ladner and Nancy A. Lynch. Relativization of Questions About Log Space Computability. *Mathematical Systems Theory*, 10:19–32, 1976.

[LV92] Ming Li and Paul M. B. Vitányi. Average Case Complexity Under the Universal Distribution Equals Worst-Case Complexity. *Inf. Process. Lett.*, 42(3):145–149, 1992.

[LV08] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications, Third Edition*. Texts in Computer Science. Springer, 2008.

[Mas79] William J Masek. Some NP-complete set covering problems. *Unpublished manuscript*, 1979.

[Mic04] Daniele Micciancio. Almost Perfect Lattices, the Covering Radius Problem, and Applications to Ajtai's Connection Factor. *SIAM J. Comput.*, 34(1):118–169, 2004.

[MR07] Daniele Micciancio and Oded Regev. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[MW17] Cody D. Murray and R. Ryan Williams. On the (Non) NP-Hardness of Computing Circuit Complexity. *Theory of Computing*, 13(1):1–22, 2017.

[NN93] Joseph Naor and Moni Naor. Small-Bias Probability Spaces: Efficient Constructions and Applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[OPS18] Igor Carboni Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:158, 2018.

[OS17] Igor Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 18:1–18:49, 2017.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. Hardness Magnification for Natural Problems. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 65–76, 2018.

[Ost91] Rafail Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. In *Proceedings of the Structure in Complexity Theory Conference*, pages 133–138, 1991.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-Way Fuctions are Essential for Non-Trivial Zero-Knowledge. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 3–17, 1993.

[Pat08] Mihai Patrascu. Succincter. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 305–313, 2008.

[PSZ00] Ramamohan Paturi, Michael E. Saks, and Francis Zane. Exponential lower bounds for depth three Boolean circuits. *Computational Complexity*, 9(1):1–15, 2000.

[PV88] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.

[Raz87] Alexander Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[RR97] Alexander A. Razborov and Steven Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[RRV02] Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the Randomness and Reducing the Error in Trevisan's Extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002.

[RS98] Alexander Russell and Ravi Sundaram. Symmetric Alternation Captures BPP. *Computational Complexity*, 7(2):152–162, 1998.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Rud97] Steven Rudich. Super-bits, Demi-bits, and NP/qpoly-natural Proofs. In *Proceedings of the Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 85–93, 1997.

[San09] Rahul Santhanam. Circuit Lower Bounds for Merlin–Arthur Classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009.

[Sho99] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Review*, 41(2):303–332, 1999.

[Sla97] Peter Slavík. A Tight Analysis of the Greedy Algorithm for Set Cover. *J. Algorithms*, 25(2):237–254, 1997.

[Smo87] Roman Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 77–82, 1987.

[Sri03] Aravind Srinivasan. On the approximability of clique and related maximization problems. *J. Comput. Syst. Sci.*, 67(3):633–651, 2003.

[STV01] Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[Tal14] Avishay Tal. Shrinkage of De Morgan Formulae by Spectral Techniques. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 551–560, 2014.

[Tor91] Jacobo Torán. Complexity Classes Defined by Counting Quantifiers. *J. ACM*, 38(3):753–774, 1991.

[Tor04] Jacobo Torán. On the Hardness of Graph Isomorphism. *SIAM J. Comput.*, 33(5):1093–1108, 2004.

[Tra84] 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.

[Tre01a] Luca Trevisan. Extractors and pseudorandom generators. *J. ACM*, 48(4):860–879, 2001.

[Tre01b] Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 453–461, 2001.

[Tur36] Alan M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[TV07] Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007.

[Uma01] Christopher Umans. The Minimum Equivalent DNF Problem and Shortest Implicants. *J. Comput. Syst. Sci.*, 63(4):597–611, 2001.

[Vad06] Salil P. Vadhan. An Unconditional Study of Computational Zero Knowledge. *SIAM J. Comput.*, 36(4):1160–1214, 2006.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3):1–336, 2012.

[Vio05] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005.

[Vol99] Heribert Vollmer. *Introduction to Circuit Complexity - A Uniform Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1999.

[Wil13] Ryan Williams. Improving Exhaustive Search Implies Superpolynomial Lower Bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

[Wil14] Ryan Williams. Nonuniform ACC Circuit Lower Bounds. *J. ACM*, 61(1):2:1–2:32, 2014.

[Yao82] Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

[Yap83] Chee-Keng Yap. Some Consequences of Non-Uniform Conditions on Uniform Classes. *Theor. Comput. Sci.*, 26:287–300, 1983.