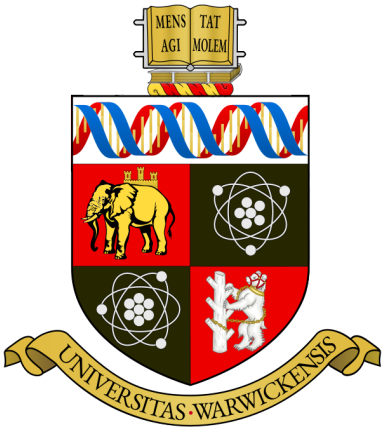


Complexity Theory of Classical and Quantum Computational Devices

by

Bruno Pasqualotto Cavalar

Thesis submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Computer Science

March 2024

Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisor Igor Oliveira for his earnest dedication and guidance through all these years. Igor was extremely generous with his time and resources, willing to offer help and advice on both my technical queries as well as my most banal ones. He always went out of his way to make sure I had the best possible research opportunities, and was consistently confident in my abilities – being a constant source of encouragement. His seemingly endless interest and knowledge in all things complexity theory was also a source of inspiration.

A thank you to my co-advisors Tom Gur, Torsten Mütze and Sayan Bhattacharya, who provided encouragement and insight in our annual meetings. I also wish to thank my examiners Christian Ikenmeyer, Srikanth Srinivasan and Matthias Englert for agreeing to help with this final stage.

I'm grateful to my coauthors Zhenjian Lu, Eli Goldin, Matthew Gray, Peter Hall, Yanyi Liu and Angelos Pelecanos for the fun discussions and for working together with me in some of the work composing this thesis. During my studies, I benefited from the lively research environment of the Simons Institute for the Theory of Computing during the Meta-Complexity program, and from the wonderful hospitality of Mika Göös at EPFL and Susanna Rezende at Lund and Copenhagen. I also wish to thank my colleagues at the Complexity Network UK, in whose meetings many of these ideas were first presented, and Arkadev Chattopadhyay for important early technical conversations.

My time at Warwick was coloured by the friendship of Marcel, Ninad, Sathya, Ian and Henry, whom I thank for making these years cheerful. A special thanks to Marcel and Thamiris for helping with our move to Coventry – settling in during the

lockdown was much easier due to their help, and made it possible for us to quickly feel at home.

It was an honour to serve with the folks at the Christian Postgraduates and Staff Network at Warwick (CPS). Thanks to Anja and Tim for organising the Forming a Christian Mind Fellows Programme in Cambridge – during that time I received a rare Christian perspective on scholarship, which I will carry through the years. Thanks to Martin Herdegen for inviting me to this world and for his continued friendship.

My time in the UK wouldn't have been nearly as enjoyable were it not for the love of my brothers and sisters at Solihull Presbyterian Church. I thank our presbyters Stephen, Falko and Jonny for their careful oversight.

I thank my parents for their persistent support and prayers for this work. My dear wife is most of all to be thanked for her love and resilience through these sometimes difficult years. She continued to believe in me and in the significance of this work even when I felt discouraged.

Above all flesh I thank the Creator for carefully hiding the secrets of theoretical computing – not too deep that we couldn't find them, nor too shallow that we would have no joy in pursuing them. It is my duty and my joy to praise Him for His glory, and with His help I will do so to the end of my days.

Declaration

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work presented was carried out by the author.

Parts of the thesis are joint work which have been published in peer-reviewed conferences and journals, all with substantial work carried out by the author:

1. **Algorithms and Lower Bounds for Comparator Circuits from Shrinkage** (Chapter 3)
   Bruno P. Cavalar, Zhenjian Lu
   Proc. 13th Innovations in Theoretical Computer Science Conference (ITCS), LIPIcs, Vol. 215, 34:1–34:21
   Algorithmica, 85(7):2131–2155, 2023

2. **Constant-Depth Circuits vs. Monotone Circuits** (Chapter 4)
   Bruno P. Cavalar, Igor Carboni Oliveira
   Proc. 38th Computational Complexity Conference (CCC), LIPIcs, Vol. 264, 29:1–29:37

Chapter 5 is based on a unpublished preprint of the author, also with substantial work carried by the author:

3. **The Computational Hardness of Quantum One-wayness** (Chapter 5)
   Bruno P. Cavalar, Eli Goldin, Matthew Gray, Peter Hall, Yanyi Liu, Angelos Pelecanos
   arXiv, 2023, https://arxiv.org/abs/2312.08363

Abstract

This thesis investigates various computational devices from the perspective of computational complexity theory, a scientific discipline concerned with the intrinsic limits and possibilities of algorithms. Most of our results are new state-of-art *complexity lower bounds*, which are mathematical proofs that certain kinds of computational devices do not exist.

Our first investigation is on *comparator circuits*. We show the first average-case lower bounds for comparator circuits, which are a significant strengthening of worst-case circuit lower bounds. Our technical study of comparator circuits also enables us to develop, for the first time, algorithms that analyse comparator circuits, such as satisfiability algorithms, pseudorandom generators and learning algorithms.

We then discuss the relative power of constant-depth circuits and monotone circuits, a very basic and fundamental question in circuit complexity. We show that there exist monotone problems that can be computed by constant-depth circuits, whereas no monotone *formula* of feasible size can compute them, extending a classical result of Okol'nishnikova (1982) and Ajtai and Gurevich (1987). We also show that, when augmented with *parity gates*, constant-depth circuits are stronger than monotone circuits, coming close to solving a conjecture of Grigni and Sipser (1992). Our study ends with a careful analysis of current techniques which demonstrates their inability to improve our results.

In our latter investigations, we switch to quantum computation, and investigate *one-way state generators*. We show a tighter connection between this primitive and pseudorandom states, and we prove that if one-way state generators exist, then a large class of natural problems (such as computing the permanent of matrices) cannot be solved efficiently by quantum algorithms. This result presents the first complexity-theoretic consequence from the existence of one-way state generators. We also prove that a fixed-copy variant of one-way state generators exists *unconditionally*, irrespective of any complexity lower bound.

It is the glory of God to conceal a matter, but the glory of kings is to search out a matter.

Proverbs 25:2

Chapter 1

Introduction

The overarching goal of computational complexity theory is to obtain a firm grasp over what makes computation *hard*. This question can be relativised with respect to various types of resources and computational models. Some of these are (or are conjectured to be) strictly more powerful than others, whereas some can be shown (or given evidence) to be incomparable. The brief history of complexity theory has also demonstrated that the *complexity-theoretic* study of such models and resources, more often than not, leads to algorithmic ideas in various fields[1].

In this thesis, we will investigate the phenomenom of computational hardness from the perspective of various computational models. We will also have a chance to observe how these ideas can be used in algorithm design and quantum cryptography. We will concern ourselves with the following types of questions:

1. What kinds of computational problems are hard?
2. Which algorithmic insights can be extracted from hardness results for computational models?
3. What is the relative power of different computational models?
4. Is complexity *necessary* for computational applications?

We will now give a brief summary of the computational models we will study and how the things we will discover about them answer in specific ways the very broad questions asked above. Later in this chapter we will introduce the models more formally and explain the results with more technical details, but for now we aim to give a big picture view of our work and the main concepts involved.

[1] See [HO02] on how algorithmic ideas are often hidden inside complexity results.

1.1 Introductory overview of the results

We begin with a very brief review of circuit models and some classical results about them. The reader unfamiliar with this material is recommended to consult [Juk12] for more details.

Boolean circuits: size and depth. The most basic and paradigmatic model of computation we will concern ourselves with is that of *Boolean circuits*. A Boolean circuit is a directed acyclic graph, in which the sources are labelled with literals $x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$, and each of the remaining vertices is labelled with an element of a fixed set of logical operations (typically the logical operators $\{\vee, \wedge\}$).[2] The source vertices are called *inputs*, as they represent naturally the n-bit binary input that is being received by the circuit, and the other vertices are called *gates*. The sink vertices (or, as in most cases in this work, *the* sink vertex) represent the outcome of the computation that logically follows from the input vertices (sources). We will typically consider that each gate has *fan-in* (that is, in-degree) constant, but sometimes we may allow unbounded fan-in – we will be explicit about which case.

One important technical distinction between general algorithms (by which we mean Turing machines) and circuits is that the latter have a *fixed* input size. As a consequence, when we say that a circuit or circuit class solves a problem L, we formally mean that there exists a *sequence* of circuits $C_1, C_2, \ldots$, one for each input length, such that $C_n(x) = 1 \iff x \in L$ for all $x \in \{0,1\}^n$. The fact that we allow one different circuit for each input length is called *non-uniformity*. When the circuit C_n can be computed in polynomial-time given n (in unary), we say that the circuit sequence is *uniform*. The distinction between uniform and non-uniform circuits, however, will rarely be important in this work, as most of our algorithms will be uniform, and most of our lower bounds will also hold for the non-uniform variants of the corresponding circuit models. Whenever our results deviate from this pattern, we will explicitly note it.

Boolean circuits have the advantage (over Turing machines) of representing algorithms with an object as concrete as a finite graph, naturally lending themselves to different complexity parameters to be studied, as well as natural restrictions on the model. Perhaps the most basic parameter one can study about circuits is *circuit size*, which is equal to the number of vertices in the circuit. Conveniently, it is a

[2] Note that it may not be necessary in general to have more than one input labelled with the same literal. However, if we require every input vertex to have bounded degree, then it may be necessary to repeat the labels.

classical result in computational complexity that one does not lose much when going from *time complexity* of Turing machines to circuit size – any algorithm that runs in time T can be implemented by a circuit of size $O(T \log T)$. Conversely, uniform polynomial-size circuits can do nothing that polynomial-time algorithms cannot.

Another significant parameter we will study is *circuit depth*, which is the length of the longest path from a source to a sink. Circuit depth allows us to define a prominent family of complexity classes known as the NC hierarchy, which we will have the opportunity to study in this work. A circuit will be called an NC^i circuit if it has polynomial-size and depth $O(\log n)^i$. Importantly, we require the fan-in to be constant. This model corresponds to *parallel* algorithms, where the depth is the runtime of the algorithm, here running on a polynomial number of processors. Of course, the class NC of problems solvable by an NC^i circuit for some fixed i can also be solved by polynomial-size circuits.

The world between NC^1 and NC^2. The first two levels of this hierarchy are already very interesting. It is known that NC^1 circuits are equivalent to polynomial-size *formulas*, which are circuits with the restrictions that every gate and every input literal has fan-out at most 1 (*fan-out* is the outdegree of the vertex). Though this circuit model is capable of solving a host of arithmetic problems [CDL01], it's conjectured that it is not able to solve problems as simple as deciding if a directed graph is connected.

The second level of the NC hierarchy, however, is surprisingly powerful. Between NC^1 and NC^2 are found a number of *space-bounded* complexity classes, such as polynomial-size *nondeterministic branching programs* and *span programs*, which are capable of solving connectivity problems on graphs and solve linear systems modulo a prime field, respectively. These computational models can be defined as follows. A *nondeterministic branching program* is a Boolean circuit with the restriction that, in each $\wedge$-gate, at least one of the incoming wires is connected to an input. Given a field $\mathbb{F}$, a $\mathbb{F}$-*span program* is a matrix M with entries in $\mathbb{F}$ where each of the rows are labelled with a literal. The matrix is said to accept an input x if the rows of M whose labels are consistent with x span the vector $\mathbb{1}$. The size of the span program is the number of rows of M. Finally, we remark that circuits of the NC^2 type are also capable of computing the determinant of a rational matrix [Mul87]. (See [Coo85] for an exploration of some complexity classes and problems between NC^1 and NC^2.)

Comparator circuits and the power of parallelism. How powerful is parallelism? In the arithmetic world[3], it is known that arbitrary polynomial-size arithmetic circuits can have their depth reduced to $O(\log^2 n)$ with only a polynomial blow-up in the size [VSBR83]. In other words, we have $\mathsf{P} = \mathsf{NC}^2$ in the algebraic world. A similar result is not known for Boolean circuits, and in fact we do not expect all polynomial-size circuits to be parallelisable.

Perhaps the weakest circuit model below P which we have evidence for not being parallelisable is that of *comparator circuits*. Comparator circuits are a powerful circuit model corresponding to computations that can be represented as a sequence of comparisons between numbers. Because of their inherently sequential nature, they are conjectured to be unparallelisable [MS92] – indeed, oracle separations between NC and comparator circuits have been found [CFL14].

As we shall see discuss in more details in Section 1.2 below, comparator circuits can be equivalently seen as Boolean circuits with fan-out 2, which immediately gives us that comparator circuits can simulate NC^1 (since, as we noted above, NC^1 circuits are equivalent to polynomial-size circuits of fan-out at most 1). More surprisingly, comparator circuits can also simulate nondeterministic branching programs, and solve a host of other problems for which no parallel algorithms are known. Comparator circuits, then, seem to pick up from NC^1 but follow a different trajectory, incomparable to that of the NC hierarchy (see Figure 1.1).

Even though comparator circuits have been studied since the work of Mayr and Subramanian [MS92], only recently have lower bounds been shown for this model [RPRC16, GR20]. These results, however, are all *worst-case* lower bounds, and do not rule out the possibility that small comparator circuits are able to give the right answer in a large proportion of the inputs. Such stronger lower bounds are called *average-case* lower bounds. More generally, the study of *average-case* hardness is a vibrant strand of complexity theory [BT06], with numerous applications (e.g., cryptography [Hir22a]).

Our work in Chapter 3, which we will summarise in detail soon below (Section 1.2), will give the first average-case lower bounds for comparator circuits matching worst-case bounds [GR20]. We will also be able to show optimal average-case lower bounds for a large family of circuit models between NC^1 and NC^2, for which

[3]Discussing arithmetic circuit complexity is beyond the scope of this work, and we mention it here simply to give a larger context of our results. For the sake of this introduction, it suffices to say that arithmetic circuits are circuits working on a fixed field $\mathbb{F}$ with multiplication and addition gates. We interpret each gate as computing a multivariate polynomial over $\mathbb{F}$. Here, one is interested in knowing which polynomials one can represent with efficient circuits (e.g., polynomial-size or polylogarithmic depth). We point the reader to [SY10] for a more detailed introduction to the topic.

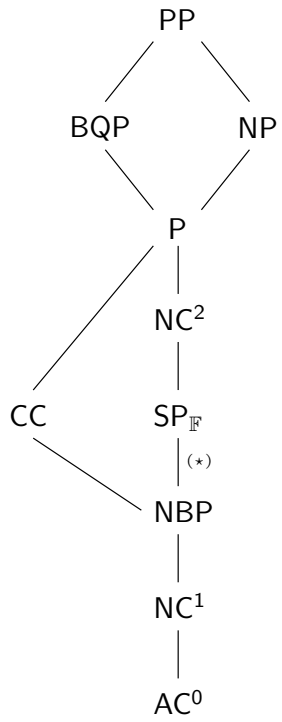


Figure 1.1: Main *non-monotone* circuit and complexity classes studied and their relationships. We denote polynomial-size comparator circuits by CC. Moreover, we denote nondeterministic branching programs by NBP, and $\mathsf{SP}_{\mathbb{F}}$ denotes $\mathbb{F}$-span programs. Uniformly, these circuit classes are also known as NL (*nondeterministic logspace*) and Mod$\mathbb{F}$-L (*logspace modular*), respectively. The containment $(\star)$ is only known to hold non-uniformly [Wig94, BG99], though NBPs are simulatable by NC^2 circuits also in the uniform case. All the other containments are known to hold both in the uniform and non-uniform cases. We denote by BQP the class of decision problems solvable by polynomial-size quantum circuits. The complexity class PP is defined in Section 1.4.3; $\mathsf{BQP} \subseteq \mathsf{PP}$ is proved in [ADH97]. The relationship between NP and BQP is unknown.

no *optimal* average-case lower bounds were known before. We will also give worst-case lower bounds on comparator circuits computing MCSP, a Boolean function of central importance in complexity theory, thus expanding the class of functions for which we can show comparator circuit lower bounds. These results will form an answer to Question 1 above, relativised to comparator circuits.

With our techniques, we will also obtain satisfiability algorithms, pseudorandom generators and learning algorithms for comparator circuits, addresing Question 2 above. We remark that those algorithms deal with some of the most fundamental problems in computer science. Satisfiability algorithms are of significant importance to practical applications [BHvMW21], and indeed most algorithms used in practice are developed for CNFs, a significantly weaker circuit model. Moreover, the construction of pseudorandom generators is a crucial step in the fundamental

theoretical question of derandomisation.

Constant-depth circuits and monotone circuits. When we introduced the NC hierarchy, we did not talk about NC^0 circuits. That's for the obvious reason that, being of constant-depth and constant fan-in, NC^0 circuits cannot compute functions that depend on all of the inputs, so they are quite limited with respect to decision problems. However, it's possible to speak meaningfully of constant-depth circuits by allowing unbounded fan-in gates – this 'extremely parallel' circuit class is known as AC^0.

It's well known that AC^0 circuits also have severe limitations of their own. They are not even able to compute the parity of its input bits – in fact, it's possible to show *exponential* size lower bounds for constant-depth, unbounded fan-in circuits that compute Parity [Hås86]. Many lower bounds and other results have been proved for AC^0 since then, making this one of the most well-understood circuit classes. It's also not hard to see that $\mathsf{AC}^0 \subseteq \mathsf{NC}^1$.

Alongside AC^0, another circuit class that has received a lot of attention throughout the years is that of *monotone circuits*, which are circuits without negation ($\neg$) gates. Exponential lower bounds are known for monotone circuits [AB87], and, for subclasses of monotone circuits (like monotone formulas), nearly best-possible lower bounds are known [PR17]. However, despite monotone computation being well-understood from this perspective, little is known about the relationship between AC^0 circuits and monotone circuits.

In Chapter 4, we will address Questions 1 and 3 above as we investigate the very basic question about the relative power of constant-depth circuits and monotone circuits, first asked by Grigni and Sipser [GS92]. As we will describe below in Section 1.3, we will be able to show, for the first time, that constant-depth circuits extended with $\oplus$-gates can solve problems that polynomial-size monotone circuis cannot, coming close to solve a conjecture of Grigni and Sipser [GS92]. We will also be able to vastly extend a classical result of Okol'nishnikova [Oko82] and Ajtai and Gurevich [AG87], by proving that AC^0 circuits are more powerful than monotone *formulas* of quasipolynomial size. Lastly on that direction, we will show surprising consequences from assuming that better separations between monotone and non-monotone circuits are impossible. One such surprising consequence will be the *collapse* of the NC hierarchy.

Our discussion in Chapter 4 will also demonstrate that, within the world between NC^1 and NC^2, a 'hardness transition' seems to occur: whereas NC^2 contains problems that monotone circuits require exponential size to compute [GKRS19], no

such problems are known within NC^1. Bringing such lower bounds closer to NC^1 is one of the main technical challenges left open by our work.

Our work will also give the first thorough investigation of the monotone complexity of Boolean *Constraint Satisfaction Problems* (CSPs), proving monotone correspondents of a classical result of Schaefer [Sch78]. The relevance of this result for the previous discussion is that most of the monotone circuit lower bounds obtained via *lifting theorems* – a prominent technique in monotone circuit complexity – also apply to this class of problems (see, e.g., [dRGR22, GKRS19]). Our analysis will show that the transition mentioned above cannot be overcome via CSPs: every CSP that is hard for monotone circuits also requires the power of non-monotone span programs to be computed. We remark that CSPs are a generalisation of the NP-complete Satisfiability problem, and they have been, and still are, the subject of much investigation, both in theory and in practice [RvBW06]. From a theoretical point of view, a major highlight is the recent discovery that that every CSP (Boolean or not) is either NP-complete or solvable in polynomial time [Zhu17, Bul17], a culmination of a long line of works beginning with Schaefer [Sch78].

Though we will mostly focus on the relationship between constant-depth circuits and general monotone circuits, we remark that a separation between comparator circuits and *monotone* comparator circuits is known [RPRC16].

The complexity of quantum one-wayness In our last technical chapter (Chapter 5), we will switch gears slightly to consider *quantum algorithms*. Quantum computing is a strenghtening of randomised computation that allows computation to evolve by means of *unitary* transformations [For03]. Conveniently, this can also be represented with a circuit model, where the gates are labelled by an element of a fixed set of unitary matrices. The complexity class of problems solvable by uniform polynomial-size quantum circuits is denoted by BQP (also denoted *quantum polynomial time*). Quantum computers carry the promise of efficiently computing more than we expect to be able to do with classical computers. One prominent such example is that of factoring integers [Sho97] – it's possible to solve this task in quantum polynomial time, whereas it's conjectured that no polynomial-time classical algorithm can do the same. Indeed, the security of the RSA cryptographic system depends on the hardness of that task, which has motivated the search for cryptographic protocols that are safe against quantum attacks, possibly based also on quantum complexity-theoretic assumptions.

A fundamental cryptographic task for classical computations is that of *one-way functions*, corresponding to easy-to-compute but hard-to-invert computations.

It's easy to show that, if one-way functions exist, then NP does not admit polynomial-time randomised algorithms (i.e., $\mathsf{NP} \not\subseteq \mathsf{BPP}$). In addition, a long list of classical results show an equivalence between one-way functions and other cryptographic and computational tasks, such as pseudorandom generators [HILL99], pseudorandom functions [GGM84], and private-key encryption [GM84]. More recently, complexity-theoretic *characterisations* have been found for the existence of one-way functions [LP20, IRS21, HIL+23]. The complexity-theoretic landscape of quantum one-wayness is, however, much less understood.

In Chapter 5, we will study a cryptographic primitive called *one-way state generators*, which has been recently proposed [MY22b] (see Section 1.4 below for a more detailed overview to our results). We will first demonstrate tighter connections between this primitive and *pseudorandom states*, a quantum analogue of pseudorandom generators. We will employ those connections to demonstrate that, when the adversary is only given a limited number of copies of the output, one-way state generators exist *unconditionally*. No analogous result is known for classical computation; this is due to the fact that, in quantum algorithms, it's impossible to *copy* states.

In our final result, we will be able to show the first complexity-theoretic consequences from the existence of one-way state generators. We will show that, if one-way state generators exist, then the permanent of integer matrices cannot be computed by polynomial-time quantum algorithms (more generally, we obtain $\mathsf{PP} \not\subseteq \mathsf{BQP}$), thus answering Question 4 for this cryptographic primitive. Unlike the previous chapters, where unconditional lower bounds for the corresponding computational models were given, the generality and power of quantum algorithms makes it very difficult for current techniques to show unconditional lower bounds. Our result will show that, based on cryptographic assumptions, it's at least possible to show the intractability of relevant computational tasks for this model. We remark that we will only be concerned with *uniform* quantum circuits in this work, and our lower bounds therefore will also hold only for this model.

We will now give a more detailed outline of each our main results separately. In Chapter 2, we will describe the main techical ingredients that compose our results. Finally, in Chapter 6, we will conclude by describing a few directions and open problems that could build on this work.

1.2 Algorithms and lower bounds for comparator circuits

We now describe our work in Chapter 3, which will be concerned with *comparator circuits*. A comparator circuit is a Boolean circuit whose gates are *comparator gates*, each of which maps a pair of inputs (x, y) to $(x \wedge y, x \vee y)$, and whose inputs are labelled by a literal (i.e., a variable x_i or its negation $\neg x_i$). We also allow constants 0 and 1 in the circuit. A convenient way of representing a comparator circuit is seen in Figure 1.2. We draw a set of horizontal lines, each of which is called a *wire* and is labelled by an input literal. The gates are represented as a sequence of vertical arrows, each of which connects some wire to another. The tip of the arrow is the logical disjunction gate ($\vee$), and the rear of the arrow is the logical conjunction gate ($\wedge$). One of the wires is selected to represent the Boolean value of the computation. The *size* of the circuit is the number of gates in the circuit. A more formal definition is given in Section 3.1.1 of Chapter 3.

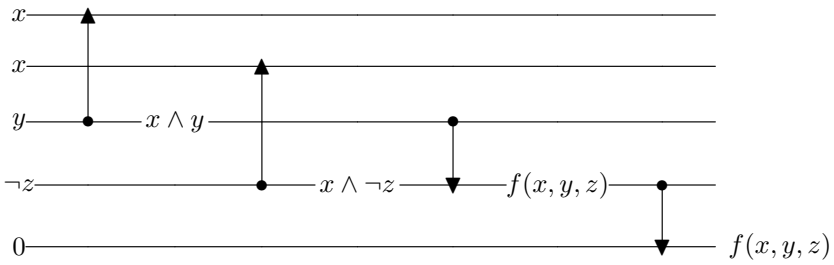


Figure 1.2: A comparator circuit with 3 inputs, 5 wires and 4 gates. The last wire computes $f(x, y, z) = (x \wedge y) \vee (x \wedge \neg z)$. We note that the same function could be computed with less wires if we removed the last wire (labelled with 0), but we decided to keep this inefficient construction for the sake of presenting a comparator circuit with full generality.

Comparator circuits can be viewed as a restricted type of circuit in which the gates have fan-out exactly two. It is easy to see that comparator circuits can efficiently simulate Boolean formulas over $\{\wedge, \vee, \neg\}$ with no overhead[4], and it's also possible to show that comparator circuits can simulate Boolean formulas over the full binary basis with only a constant factor blow-up in the size (we give a full proof of this observation in Appendix A.1). Moreover, it is also known that polynomial-size comparator circuits can even simulate nondeterministic branching programs [MS92] with a polynomial overhead only. On the other hand, compara-

[4] As a comparison, note that there are linear-size comparator circuits for Parity (see Figure 1.3 or Theorem A.1.2), whereas any Boolean formula computing Parity has size $\Omega(n^2)$ [Hra71].

tor circuits appear to be much stronger than formulas[5], as it is conjectured that polynomial-size comparator circuits are incomparable to NC [MS92]. Evidence for this conjecture is that polynomial-size comparator circuits can compute problems whose known algorithms are inherently sequential, such as stable marriage and lexicographically first maximal matching [MS92], and there is an oracle separation between NC and polynomial-size comparator circuits [CFL14]. Moreover, Robere, Pitassi, Rossman and Cook [RPRC16] showed that there exists a Boolean function in mNC^2 not computed by polynomial-size *monotone* comparator circuits[6]. For these reasons, comparator circuits are likely to be incomparable to NC, and polynomial-size span programs, which are contained in NC^2, are not expected to be stronger than polynomial-size comparator circuits.

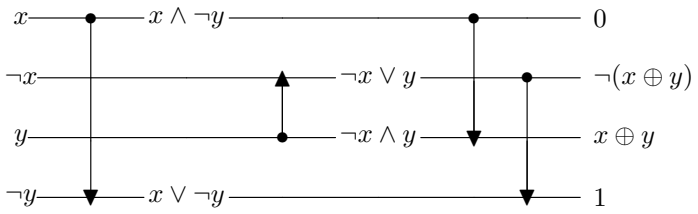


Figure 1.3: A comparator circuit computing $\oplus, \neg\oplus, 0, 1$.

Despite the importance of comparator circuits, we don't know much about them. Though it is easy to see that Parity can be computed by comparator circuits with $O(n)$ wires and gates (see Figure 1.3 or Appendix A.1), the best known comparator circuit for Majority uses $O(n)$ wires and $O(n \log n)$ gates [AKS83]. We don't know if there is a linear-size comparator circuit for Majority[7], whereas, for the weaker model of nondeterministic branching programs, superlinear lower bounds are known [Raz90]. Structural questions about comparator circuits have also received some attention in recent years [GKRS19, KSS18].

The first superlinear *worst-case lower bound* for comparator circuits was recently obtained by Gál and Robere [GR20], by an adaptation of Nečiporuk's argument [Nec66]. Their proof yields a lower bound of $\Omega\big((n/\log n)^{1.5}\big)$ to comparator circuits computing a function of n bits. For *monotone* comparator circuits, exponential lower bounds are known [RPRC16].

In Chapter 3, we exploit structural properties of small-size comparator circuits in order to prove the first *average-case lower bounds* and design the first *circuit*

[5]Recall that the class of polynomial-size formulas is exactly NC^1.

[6]Comparator circuits are monotone if they don't have negated literals.

[7]As opposed to a sorting network, note that a comparator circuit can use multiple copies of the same input literal.

analysis algorithms for small comparator circuits. Developing circuit analysis algorithms is a crucial step for understanding a given circuit class [Oli13, Wil14a], and are often obtained only after lower bounds have been proven for the class[8]. Many well-studied circuit classes have been investigated from this perspective, such as AC^0 circuits [IMP12], De Morgan formulas [Tal15], branching programs [IMZ19], ACC circuits [Wil14b], and many others (see also [CKK+15, ST17, KKL+20]). Our work commences an investigation of this kind for comparator circuits.

1.2.1 Average-case lower bounds

Chapter 3 begins with the first average-case lower bound against comparator circuits.

Theorem 1.2.1 (Average-case Lower Bound)**.** *There exist constants $c, d \geqslant 1$ such that the following holds. For any $k \geqslant c \cdot \log n$, there is a polynomial-time computable function f_k such that, for every comparator circuit C with at most*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}}$$

gates, we have

$$\mathop{\mathbb{P}}_{x \in \{0,1\}^n}[f_k(x) = C(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}}.$$

An important feature of the lower bound in Theorem 1.2.1 is that it matches the $\Omega\big((n/\log n)^{1.5}\big)$ worst-case lower bound of [GR20], in the sense that we can recover their result (up to a multiplicative constant) by setting $k = O(\log n)$.

Using ideas from the proof of the above average-case lower bound, we also show average-case lower bounds against various models that tightly match their state-of-the-art worst-case lower bounds, such as general formulas, (deterministic-, nondeterministic-, parity-)branching programs and span programs (see Section 3.3). Note that strong average-case lower bounds against $n^{2-o(1)}$-size general formulas and deterministic branching programs were previously known [KR13, CKK+15] but they did not match the worst-case lower bounds, whereas tight average-case lower bounds for *De Morgan* formulas were proved by [KRT17].

[8] One exception is ACC circuits, for which satisfiability algorithms are known [Wil14b], and the only exponential lower bound known for ACC is a consequence of this algorithm. However, the function used in the lower bound is not in NP.

1.2.2 Algorithms for comparator circuits

We now describe the algorithms we can design based on our techniques for proving lower bounds for comparator circuits.

#SAT algorithms. The design of algorithms for interesting *circuit analysis problems* is a growing line of research in circuit complexity [Wil14a]. These are problems that take circuits as inputs. A famous example of such a circuit analysis problem is the *satisfiability* problem (SAT), which asks to determine whether a given circuit has a satisfying assignment. Note that the satisfiability problem for polynomial-size general circuits is NP-complete, so it is not believed to have a polynomial-time (or subexponential-time) algorithm. However, one can still ask whether we can obtain *non-trivial* SAT algorithms running faster than exhaustive search, say in time $2^n/n^{\omega(1)}$ where n is the number of variables of the input circuit, even for restricted circuit classes. While designing non-trivial SAT algorithms is an interesting problem by itself, it turns out that this task is also tightly connected to proving lower bounds. In particular, recent works of Williams [Wil13, Wil14b] have shown that such a non-trivial satisfiability algorithm for a given class of circuits can often be used to prove non-trivial circuit lower bounds against that same circuit class.

Here, we show an algorithm with non-trivial running time that counts the number of satisfying assignments of a given comparator circuit.

Theorem 1.2.2 (#SAT Algorithms)**.** *There is a constant $d > 1$ and a deterministic algorithm such that, for every $k \leqslant n/4$, given a comparator circuit on n variables with at most*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}}$$

gates, the algorithm outputs the number of satisfying assignments of C in time

$$2^{n-k} \cdot \mathsf{poly}(n).$$

Note that the running time in Theorem 1.2.2 is non-trivial for size up to $o(n/\log n)^{1.5}$, in which case $k = \omega(\log n)$ and the running time becomes $2^n/n^{\omega(1)}$.

Pseudorandom generators and MCSP lower bounds. Another important circuit analysis problem is *derandomisation*, which, roughly speaking, asks to decide whether a given circuit accepts or rejects a large fraction of its inputs. A standard approach to solve this problem is to construct a pseudorandom generator (PRG).

A PRG against a class $\mathcal{C}$ of circuits is an efficient and deterministic procedure G mapping short binary strings (seeds) to longer binary strings, with the property that G's output (over uniformly random seeds) "looks random" to every circuit in $\mathcal{C}$. More precisely, we say that a generator $G\colon \{0,1\}^r \to \{0,1\}^n$ *ε-fools* a class $\mathcal{C}$ of circuits if, for every $C\colon \{0,1\}^n \to \{0,1\}$ from $\mathcal{C}$, we have

$$\left| \mathop{\mathbb{P}}_{z\in\{0,1\}^r}[C(G(z)) = 1] - \mathop{\mathbb{P}}_{x\in\{0,1\}^n}[C(x) = 1] \right| \leqslant \varepsilon.$$

In constructing PRGs, we aim to minimize the parameter r, which is called the seed length.

We show a PRG against comparator circuits of size s with seed length $s^{2/3+o(1)}$.

Theorem 1.2.3 (Pseudorandom Generators)**.** *For every $\varepsilon : \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ such that $\varepsilon(n) \geqslant 1/\mathsf{poly}(n)$ and $s(n) = n^{\Omega(1)}$, there exists a sequence $(G_n)_{n\in\mathbb{N}}$ of pseudorandom generators $G_n : \{0,1\}^r \to \{0,1\}^n$, with seed length*

$$r = s^{2/3+o(1)},$$

that ε-fools comparator circuits on n variables with $s(n)$ gates.

Note that the seed length of the PRG in Theorem 1.2.3 is non-trivial (i.e., $o(n)$) for comparator circuits of size $n^{1.5-o(1)}$, and only guaranteed to be smaller than the output length *for large enough* n, since our bound on the seed length is merely asymptotic[9].

The PRG above has an application in obtaining lower bounds for the *minimum circuit size problem* (MCSP) against comparator circuits. The MCSP problem asks if a given truth table[10] represents a function that can be computed by some small-size circuit. Understanding the exact complexity of MCSP is a fundamental problem in complexity theory. Motivated by a recent line of research called *hardness magnification* [OS18, OPS19, CJW19, CHO+20], which states that a weak circuit lower bound for certain variants of MCSP implies breakthrough results in circuit complexity, researchers have been interested in showing lower bounds for MCSP

[9] The theorem states that, for *every* $n \in \mathbb{N}$, there exists a generator G mapping r bits to n bits for any choice of the parameters $\varepsilon : \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ satisfying that $s(n) = n^{\Omega(1)}$ and $\varepsilon(n) \geqslant n^{-k}$ for some $k \geqslant 1$. The theorem thus guarantees that, for every n, the generator will ε-fool comparator circuits with s gates. For concrete choices of ε and s (e.g., $\varepsilon = 1/3$ and $s = n$), this is meaningful for every $n \in \mathbb{N}$. However, the guarantee on the seed-length given here is only asymptotic, which means that it is strictly smaller than the output n only for large enough $n \in \mathbb{N}$.

[10] A truth table is a bit-string that stores the output values of a Boolean function for all possible inputs.

against restricted classes of circuits. For many restricted circuit classes such as constant-depth circuits, formulas and branching programs, the lower bounds that have been proved for MCSP essentially match the best known lower bounds that we have for any explicit functions [GII+19, CKLM20, KKL+20]. Here we obtain MCSP lower bounds against comparator circuits that nearly match the worst-case lower bounds.

Theorem 1.2.4 (MCSP Lower Bounds)**.** *Let* $\mathsf{MCSP}[n^\alpha]$ *denote the problem of deciding whether a given* n*-bit truth table represents a function that can be computed by some general circuit of size at most* n^α*. For every large enough* $n \in \mathbb{N}$*, any* $\varepsilon > 0$ *and any* $0 < \alpha \leqslant 1 - \varepsilon$*, the* $\mathsf{MCSP}[n^\alpha]$ *problem does not have comparator circuits with* $n^{1+\alpha/2-\varepsilon}$ *gates.*

Previously, non-trivial comparator circuit lower bounds were known only for functions satisfying Nečiporuk's criterion [Nec66, GR20], such as *Element Distinctness* and *Indirect Storage Access*. Theorem 1.2.4 provides yet another natural computational problem which is hard for bounded-size comparator circuits. We remark that the MCSP problem is expected to require much larger circuits than the lower bound of Theorem 1.2.4 provides; however, the lack of combinatorial, algebraic or analytic structure in the MCSP function means that proving lower bounds for it is usually hard.

Finally, we also observe that the framework developed in [ST17] can be used to obtain a *non-trivial* (distribution-independent) PAC learning algorithm for comparator circuits of size $n^{1.5-o(1)}$, that uses membership queries (see Section 3.6 of Chapter 3).

1.3 Constant-depth circuits vs. monotone circuits

Chapter 4 turns our focus to *monotone* Boolean functions. A Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is monotone if $f(x) \leqslant f(y)$ whenever $x_i \leqslant y_i$ for each coordinate $1 \leqslant i \leqslant n$. Monotone Boolean functions, and the monotone Boolean circuits[11] that compute them, have been extensively investigated for decades due to their relevance in circuit complexity [Raz85a], cryptography [BL88], learning theory [BT96], proof complexity [Kra97, Pud97], property testing [GGLR98], pseudorandomness [CZ16], optimisation [GJW18], hazard-free computations [IKL+19], and meta-complexity [Hir22b], among other topics. In addition, over the last few years

[11] Recall that in a monotone Boolean circuit the gate set is limited to $\{\mathsf{AND}, \mathsf{OR}\}$ and input gates are labelled by elements from $\{x_1, \ldots, x_n, 0, 1\}$.

a number of results have further highlighted the importance of monotone complexity as a central topic in the study of propositional proofs, total search problems, communication protocols, and related areas (see [dRGR22] for a recent survey).

Some of the most fundamental results about monotone functions deal with their complexities with respect to different classes of Boolean circuits, such as the monotone circuit lower bound of Razborov [Raz85b] for Matching and the constant-depth circuit lower bound of Rossman [Ros08b] for k-Clique. Particularly important to our discussion is a related strand of research that contrasts the computational power of monotone circuits relative to general (non-monotone) AND/OR/NOT circuits, which we review next.

Weakness of Monotone Circuits. The study of monotone simulations of non-monotone computations and associated separation results has a long and rich history. In a sequence of celebrated results, [Raz85b, And85, AB87, Tar88] showed the existence of monotone functions that can be computed by circuits of polynomial size but require monotone circuits of size $2^{n^{\Omega(1)}}$. In other words, the use of negations can significantly speedup the computation of monotone functions. More recently, Göös, Kamath, Robere and Sokolov [GKRS19] considerably strengthened this separation by showing that some monotone functions in NC^2 (poly-size $O(\log^2 n)$-depth fan-in two circuits) require monotone circuits of size $2^{n^{\Omega(1)}}$. (An earlier weaker separation against monotone depth $n^{\Omega(1)}$ was established in [RW92].) Therefore, negations can also allow monotone functions to be efficiently computed in parallel.

Similar separations about the limitations of monotone circuits are also known at the low-complexity end of the spectrum: Okol'nishnikova [Oko82] and (independently) Ajtai and Gurevich [AG87] exhibited monotone functions in AC^0 (i.e., constant-depth poly-size AND/OR/NOT circuits) that require monotone AC^0 circuits (composed of only AND/OR gates) of super-polynomial size.[12] This result has been extended to an exponential separation in [COS17], which shows the existence of a monotone function in AC^0 that requires monotone depth-d circuits of size $2^{\widetilde{\Omega}(n^{1/d})}$ even if MAJ (majority) gates are allowed in addition to AND/OR gates.[13]

Strength of Monotone Circuits. In contrast to these results, in many settings negations do not offer a significant speedup and monotone computations can be un-

[12] We refer to [BST13] for an alternate exposition of this result.

[13] Separations between monotone and non-monotone devices have also been extensively investigated in other settings. This includes average-case complexity [BHST14], different computational models, such as span programs [BGW99, RPRC16] and algebraic complexity (see [CDM21] and references therein), and separations in first-order logic [Sto95, Kup21, Kup22]. We restrict our attention to worst-case separations for Boolean circuits in Chapter 4.

expectedly powerful. For instance, monotone circuits are able to efficiently implement several non-trivial algorithms, such as solving constraint satisfaction problems using treewidth bounds (see, e.g., [Oli15, Chapter 3]). As another example, in the context of cryptography, it has been proved that if one-way functions exist, then there are monotone one-way functions [GI12]. Below we describe results that are more closely related to the separations investigated in Chapter 4.

In the extremely constrained setting of depth-2 circuits, Quine [Qui53] showed that monotone functions computed by size-s DNFs (resp., CNFs) can always be computed by size-s monotone DNFs (resp., CNFs). Some results along this line are known for larger circuit depth, but with respect to more structured classes of monotone Boolean functions. Rossman [Ros08a, Ros17b] showed that any homomorphism-preserving graph property computed by AC^0 circuits is also computed by monotone AC^0 circuits.[14] Under no circuit depth restriction, Berkowitz [Ber82] proved that the monotone and non-monotone circuit size complexities of every slice function are polynomially related.[15]

Despite much progress and sustained efforts, these two classes of results leave open tantalising problems about the power of cancellations in computation.[16] In particular, they suggest the following basic question about the contrast between the weakness of monotone computations and the strength of negations:

What is the largest computational gap between the power of monotone and general (non-monotone) Boolean circuits?

A concrete formalisation of this question dates back to the seminal work on monotone complexity of Grigni and Sipser [GS92] in the early nineties. They asked if there are monotone functions in AC^0 that require super-polynomial size monotone Boolean circuits, i.e., if $\mathsf{AC}^0 \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[\mathsf{poly}]$. In case this separation holds, it would exhibit the largest qualitative gap between monotone and general Boolean circuits, i.e., even extremely parallel non-monotone computations can be more efficient than arbitrary monotone computations.

[14] A function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ is called a *graph property* if $f(G) = f(H)$ whenever G and H are isomorphic graphs, and *homomorphism-preserving* if $f(G) \leqslant f(H)$ whenever there is a graph homomorphism from G to H. It is easy to see that every homomorphism-preserving graph property is monotone.

[15] A function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ is a *slice function* if there is $i \geqslant 0$ such that $f(x)$ is 0 on inputs of Hamming weight less than i and 1 on inputs of Hamming weight larger than i.

[16] Any non-monotone circuit can be written as an XOR (parity) of distinct monotone sub-circuits (see, e.g., [GMOR15, Appendix A.1]), so negations can be seen as a way of combining, or cancelling, different monotone computations. See also a related discussion in Valiant [Val80].

Results and preliminaries. Our results show that, with respect to the computation of monotone functions, highly parallel (non-monotone) Boolean circuits can be super-polynomially more efficient than unrestricted monotone circuits. Before providing a precise formulation of these results, we introduce some notation.

For a function $d\colon \mathbb{N} \to \mathbb{N}$, let $\mathsf{mDEPTH}[d]$ denote the class of Boolean functions computed by monotone fan-in two AND/OR Boolean circuits of depth $O(d(n))$. Similarly, we use $\mathsf{mSIZE}[s]$ to denote the class of Boolean functions computed by monotone circuits of size $O(s(n))$. More generally, for a circuit class $\mathcal{C}$, we let $\mathsf{m}\mathcal{C}$ denote its natural monotone analogue. Finally, for a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, we use $\mathsf{mSIZE}(f)$ and $\mathsf{mDEPTH}(f)$ to denote its monotone circuit size and depth complexities, respectively. We refer to Jukna [Juk12] for standard background on circuit complexity theory.

1.3.1 Constant-depth circuits vs. monotone circuits

Recall that the Okol'nishnikova-Ajtai-Gurevich [Oko82, AG87] theorem states that $\mathsf{AC}^0 \cap \mathsf{Mono} \not\subseteq \mathsf{mAC}^0$. In contrast, as our main result, we establish a separation between constant-depth Boolean circuits and monotone circuits of much larger depth. In particular, we show that constant-depth circuits with negations can be significantly more efficient than monotone formulas.

Theorem 1.3.1 (Polynomial-size constant-depth vs. larger monotone depth)**.** *For every $k \geqslant 1$, we have $\mathsf{AC}^0 \cap \mathsf{Mono} \not\subseteq \mathsf{mDEPTH}[(\log n)^k]$. Moreover, this separation holds for a monotone graph property.*

In a more constrained setting, Kuperberg [Kup21, Kup22] exhibited a monotone graph property expressible in first-order logic that cannot be expressed in positive first-order logic. A separation that holds for a monotone graph property was unknown even in the context of AC^0 versus mAC^0.

Let $\mathsf{HomPreserving}$ denote the class of all homomorphism-preserving graph properties, and recall that Rossman [Ros08a, Ros17b] proved $\mathsf{AC}^0 \cap \mathsf{HomPreserving} \subseteq \mathsf{mAC}^0$. Theorem 1.3.1 implies that this efficient monotone simulation does not extend to the larger class of monotone graph properties, even if super-logarithmic depth is allowed.

Our argument is completely different from those of [Oko82, AG87, BST13, COS17] and their counterparts in first-order logic [Sto95, Kup21, Kup22]. In particular, it allows us to break the $O(\log n)$ monotone depth barrier present in previous separations with an AC^0 upper bound, which rely on lower bounds against monotone circuits of depth d and size (at most) $2^{n^{O(1/d)}}$. We defer the discussion of our

techniques to Section 2.2 in Chapter 4.

In our next result, we consider monotone circuits of unbounded depth.

Theorem 1.3.2 (Polynomial-size constant-depth vs. larger monotone size)**.** *For every $k \geqslant 1$, we have* $\mathsf{AC}^0[\oplus] \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{(\log n)^k}]$.

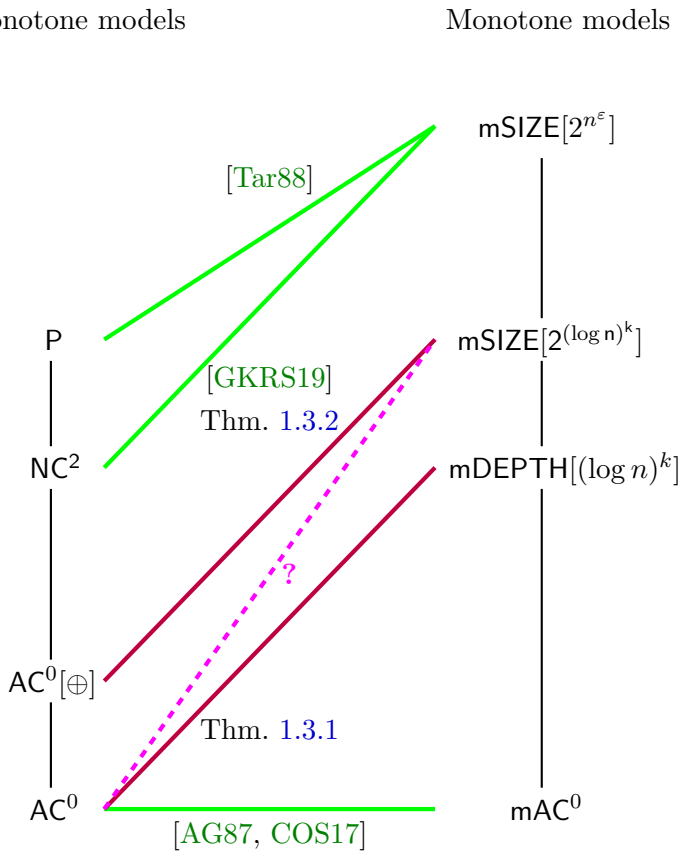


Figure 1.4: Illustration of the results of Chapter 4. The lines from left to right represent a result showing that there exists a monotone Boolean function in the left circuit class which is not computed by the monotone circuit class in the right. The green lines represent known results. The purple lines represent our results. The dashed magenta line represents a strengthening of the open question of Grigni and Sipser [GS92], corresponding to a lower bound like that of our Theorem 1.3.2, but with an upper bound like that of our Theorem 1.3.1. To make the figure simpler, we didn't draw other lines that could have been drawn, such as a green line from NC^2 to quasipolynomial monotone size ([Raz85b]), or a green line from NC^2 to monotone $n^{\Omega(1)}$-depth ([RW92]).

Theorem 1.3.1 and Theorem 1.3.2 are incomparable: while the monotone lower bound is stronger in the latter, its constant-depth upper bound requires parity gates. Theorem 1.3.2 provides the first separation between constant-depth circuits and monotone circuits of polynomial size, coming remarkably close to a solution to the question considered by Grigni and Sipser [GS92]. Indeed, for $k = 2$, our lower

bound is $n^{\Omega(\log n)}$, which is superpolynomial as Grigni and Sipser asked for, whereas the upper bound is a constant-depth circuit with $\oplus$-gates.

We note that in both of our results the family of monotone functions is explicit and has a simple description (see the technical overview in Section 2.2).

1.3.2 Non-trivial monotone simulations and their consequences

While Theorem 1.3.1 and Theorem 1.3.2 provide more evidence for the existence of monotone functions in AC^0 which require monotone circuits of superpolynomial size, they still leave open the intriguing possibility that unbounded fan-in $\oplus$-gates might be crucial to achieve the utmost cancellations (speedups) provided by constant-depth circuits. This further motivates the investigation of efficient monotone simulations of constant-depth circuits without parity gates, which we consider next.

For convenience, let $\mathsf{AC}^0_d[s]$ denote the class of Boolean functions computed by AC^0 circuits of depth $\leqslant d$ and size $\leqslant s(n)$. (We might omit $s(n)$ and/or d when implicitly quantifying over all families of polynomial size circuits and/or all constant depths.)

We observe that a non-trivial monotone simulation is possible in the absence of parity gates. Indeed, by combining existing results from circuit complexity theory, it is not hard to show that $\mathsf{AC}^0_d[s] \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{n(1-1/O(\log s)^{d-1})}]$ (see Section 4.3.1). Moreover, this upper bound is achieved by monotone DNFs of the same size. This is the best upper bound we can currently show for the class of all monotone functions when the depth d satisfies $d \geqslant 3$. (Negations offer no speedup at depths $d \leqslant 2$ [Qui53].) In contrast, we prove that a significantly faster monotone simulation would lead to new (non-monotone) lower bounds in complexity theory. Recall that it is a notorious open problem to obtain explicit lower bounds against depth-d circuits of size $2^{\omega(n^{1/(d-1)})}$, for any fixed $d \geqslant 3$. We denote by $\mathsf{GraphProperties}$ the set of all Boolean functions which are graph properties.

Theorem 1.3.3 (New circuit lower bounds from monotone simulations)**.** *There exists $\varepsilon > 0$ such that the following holds.*

1. *If $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$, then $\mathsf{NP} \not\subseteq \mathsf{AC}^0_3[2^{o(n)}]$.*
2. *If $\mathsf{AC}^0_4 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{NP} \not\subseteq \mathsf{AC}^0_4[2^{o(\sqrt{n}/\log n)}]$.*
3. *If $\mathsf{AC}^0 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$.*
4. *If $\mathsf{NC}^1 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{O(n^{\varepsilon})}]$, then $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$.*

5. *If* $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathrm{poly}]$, *then* $\mathsf{NP} \not\subseteq \mathsf{NC}^1$.

6. *If* $\mathsf{NC}^1 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathrm{poly}]$, *then* $\mathsf{L} \not\subseteq \mathsf{NC}^1$.

Item (3) of Theorem 1.3.3 implies in particular that, if the upper bound of Theorem 1.3.2 cannot be improved to AC^0 (i.e., the question asked by [GS92] has a negative answer), then $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$. It also improves a result from [CHO+20] showing the weaker conclusion $\mathsf{NP} \not\subseteq \mathsf{NC}^1$ under the same assumption.

Even if it's impossible to efficiently simulate AC^0 circuits computing monotone functions using unbounded depth monotone circuits, it could still be the case that a simulation exists for certain classes of monotone functions with additional structure. As explained above, Rossman's result [Ros08a, Ros17b] achieves this for graph properties that are preserved under homomorphisms. Items (5) and (6) of Theorem 1.3.3 show that a simulation that holds for all monotone graph properties is sufficient to get new separations in computational complexity.

1.3.3 Monotone complexity of constraint satisfaction problems

Recall that [GKRS19] showed the existence of a monotone function f^{GKRS} in NC^2 that is not in $\mathsf{mSIZE}[2^{n^{\Omega(1)}}]$. As opposed to classical results [Raz85b, And85, AB87, Tar88] that rely on the approximation method, their monotone circuit lower bound employs a lifting technique from communication complexity. It is thus natural to consider if their approach can be adapted to provide a monotone function g that is efficiently computable by constant-depth circuits but is not in $\mathsf{mSIZE}[\mathrm{poly}]$.

As remarked in [GKRS19, dRGR22], most[17] monotone lower bounds obtained from lifting theorems so far also hold for monotone encodings of constraint satisfaction problems (CSPs). Next, we introduce a class of monotone Boolean functions $\mathsf{CSP\text{-}SAT}_S$ which capture the framework and lower bound of [GKRS19].

Encoding CSPs as monotone Boolean functions. Let $R \subseteq \{0,1\}^k$ be a relation. We call k the *arity* of R. Let $V = (i_1, \ldots, i_k) \in [n]^k$, and let $f_{R,V} : \{0,1\}^n \to \{0,1\}$ be the function that accepts a string $x \in \{0,1\}^n$ if $(x_{i_1}, \ldots, x_{i_k}) \in R$. We call $f_{R,V}$ a *constraint application* of R on n variables. (A different choice of the sequence V gives a different constraint application of R.) If S is a finite set of Boolean relations, we call any set of constraint applications of relations from S on a fixed set of variables an *S-formula.* In particular, we can describe an S-formula through a set of pairs (V, R). We say that an S-formula F is *satisfiable* if there exists an assignment to the variables of F which satisfies all the constraints of F.

[17] For a more careful discussion of this, see Section 4.4.5 in Chapter 4.

Let $S = \{R_1, \ldots, R_k\}$ be a finite set of Boolean relations. Let ℓ_i be the arity of the relation R_i. Note that there are n^{ℓ_i} possible constraint applications of the relation R_i on n variables. Let $N := \sum_{i=1}^{k} n^{\ell_i}$. We can identify each S-formula F on a fixed set of n variables with a corresponding string $w^F \in \{0,1\}^N$, where $w^F_j = 1$ if and only if the j-th possible constraint application (corresponding to one of the N pairs (V, R)) appears in F. Let $\mathsf{CSP\text{-}SAT}^n_S : \{0,1\}^N \to \{0,1\}$ be the Boolean function which accepts a given S-formula F if F is *unsatisfiable*. Note that this is a monotone function. When n is clear from the context or we view $\{\mathsf{CSP\text{-}SAT}^n_S\}_{n \geqslant 1}$ as a sequence of functions, we simply write $\mathsf{CSP\text{-}SAT}_S$.

The function f^{GKRS} from [GKRS19] is simply $\mathsf{CSP\text{-}SAT}_S$ for $S = \{\oplus^0_3, \oplus^1_3\}$, where $\oplus^b_3(x_1, x_2, x_3) = 1$ if and only if $\sum_i x_i = b \pmod 2$. More generally, for any finite set S of Boolean relations, their framework shows how to lift a Resolution width (resp. depth) lower bound for an arbitrary unsatisfiable S-formula F over m variables into a corresponding monotone circuit size (resp. depth) lower bound for $\mathsf{CSP\text{-}SAT}^n_S$, where $n = \mathsf{poly}(m)$.

Despite the generality of the technique from [GKRS19] and the vast number of possibilities for S, we prove that a direct application of their approach cannot establish Theorem 1.3.1 and Theorem 1.3.2. This is formalised as follows. (We refer to Section 4.4 for much stronger forms of the result.)

Theorem 1.3.4 (Limits of the direct approach via lifting and CSPs)**.** *Let S be a finite set of Boolean relations. The following holds.*

1. *If $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mSIZE}[\mathsf{poly}]$ then $\mathsf{CSP\text{-}SAT}_S$ is $\oplus\mathsf{L}$-hard under $\leqslant^{\mathsf{AC}^0}_m$ reductions.*
2. *If $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^1$ then $\mathsf{CSP\text{-}SAT}_S$ is L-hard under $\leqslant^{\mathsf{AC}^0}_m$ reductions.*

In particular, since there are functions (e.g., $\mathsf{Majority}$) computable in logarithmic space that are not in $\mathsf{AC}^0[\oplus]$, Theorem 1.3.4 (Part 2) implies that any $\mathsf{CSP\text{-}SAT}_S$ function that is hard for poly-size monotone formulas (mNC^1) must lie outside $\mathsf{AC}^0[\oplus]$. Observe that this can also be interpreted as a *monotone simulation*: for any finite set S of Boolean relations, if $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0[\oplus]$ then $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNC}^1$.[18]

Theorem 1.3.4 is a corollary of a general result that completely classifies the monotone circuit complexity of Boolean-valued constraint satisfaction problems

[18] Jumping ahead, our proof of Theorem 1.3.2 still relies in a crucial way on the monotone lower bound obtained by [GKRS19]. However, our argument requires an extra ingredient and does not follow from a direct application of their template. We provide more details about it in our technical overview in Section 2.2. Interestingly, the proof of Theorem 1.3.1 was discovered by trying to avoid the "barrier" posed by Theorem 1.3.4.

based on the set $\mathsf{Pol}(S)$ of *polymorphisms* of S, a standard concept in the investigation of CSPs.[19] We present next a simplified version of this result, which shows a dichotomy for the monotone circuit size and depth of Boolean-valued constraint satisfaction problems. We refer to Section 4.4 for a more general formulation and additional consequences.

Theorem 1.3.5 (Dichotomies for the monotone complexity of Boolean-valued CSPs)**.** *Let S be a finite set of Boolean relations. The following holds.*

1. Monotone Size Dichotomy: *If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$*, then there exists* $\varepsilon > 0$ *such that* $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = 2^{\Omega(n^\varepsilon)}$*. Otherwise,* $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = n^{O(1)}$*.*

2. Monotone Depth Dichotomy: *If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$ *or* $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$ *or* $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$*, there is* $\varepsilon > 0$ *such that* $\mathsf{mDEPTH}(\mathsf{CSP\text{-}SAT}_S) = \Omega(n^\varepsilon)$*. Otherwise,* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNC}^2$*.*

We note that previous papers of Schaefer [Sch78] and Allender, Bauland, Immerman, Schnoor and Vollmer [ABI+09] provided a *conditional* classification of the complexity of such CSPs. Theorem 1.3.5 and its extensions, which build on their results and techniques, paint a complete and *unconditional* picture of their monotone complexity.[20]

1.4 The computational hardness of quantum one-wayness

In Chapter 5, we switch from restricted circuit models to a more powerful computational device – that of quantum algorithms – which we will study from the perspective of cryptography.

The vast majority of useful classical cryptographic primitives share the following property: they can be used to build one-way functions[21] in a black-box manner. In this sense, one-way functions can be thought of as a "minimal" cryptographic primitive. However, any one-way function can be broken by an efficient algorithm with access to an NP oracle. This means that if $\mathsf{P} = \mathsf{NP}$, then one-way functions do not exist. As it is unknown whether $\mathsf{P} = \mathsf{NP}$ or not, the existence of

[19] Roughly speaking, $\mathsf{Pol}(S)$ captures the amount of symmetry in S, and a larger set $\mathsf{Pol}(S)$ implies that solving $\mathsf{CSP\text{-}SAT}_S$ is computationally easier. We refer the reader to Section 4.4 for more details and for a discussion of Post's lattice, which is relevant in the next statement.

[20] We remark that only recently has Schaefer's classification been extended to the non-Boolean case [Zhu17, Bul17]. Though the refined classification of [ABI+09] is conjectured to hold analogously in the case of non-Boolean CSPs [LT09], this is still open (see the discussion in [Bul18, Section 7]).

[21] A one-way function is a function on bit-strings which can be efficiently evaluated but is hard to invert.

one-way functions, and thus all of classical cryptography, must rely on computational assumptions.

This issue led to the natural desire to "map out" the world of classical cryptography. Over many years, cryptographers have done a fairly good job of figuring out which cryptographic primitives can be built from each other. This cartography helps give a sense of the relative strength of assuming the existence of different cryptographic primitives.

As an example, it is known how to construct one-way functions from any key exchange protocol, i.e. a protocol where two parties can agree on a secret using only communication over a public channel [BCG89]. However, there is strong evidence that building a key exchange protocol from a one-way function is difficult [IR89]. The primitives which can be built from one-way function form a crypto-complexity class known as "MiniCrypt" [Imp95]. Two cryptographic primitives in this class of particular note are pseudorandom generators and commitment schemes. A pseudorandom generator is a deterministic function which maps a small amount of randomness to a longer string indistinguishable from random. A commitment scheme is a process by which a party can encode some string into a "commitment", such that later the party can prove this "commitment" was an encoding of the original string.

In recent years, cryptographers have started to consider what happens if we allow cryptographic primitives to have quantum output. Here, the landscape of relations between primitives looks very different. Of particular note, it was shown that quantum key distribution, a quantum variant of key exchange, exists unconditionally [BB14, Wie83]. On the other hand, it is known that the quantum versions of one-way functions, pseudorandom generators, and commitments cannot be secure against information-theoretic attackers, and thus require some computational hardness in order to exist [LC97, JLS18, KT23]. These variants are known as one-way state generators, pseudorandom state generators, and quantum bit commitments respectively.

However, it is still unclear what this hardness looks like from a complexity perspective. In particular, it is known there exists an oracle relative to which $\mathsf{BQP} \supseteq \mathsf{NP}$, but all three of these primitives exist[22] [Kre21]. Furthermore, we are still mapping out the relations between quantum primitives. It was only recently discovered that quantum bit commitments can be built from one-way state generators [KT23], and it is still an open question as to whether pseudorandom state

[22]For one-way state generators and quantum bit commitments, the result follows from [Kre21] and the subsequent works of [MY22b, KT23].

generators can be built from quantum bit commitments.

The main goal of Chapter 5 is to broaden our understanding of the hardness of quantum primitives, with a particular focus on one-way state generators. In particular, we show two main results:

1. One-way state generators can be built from pseudorandom states for nearly all parameter regimes requiring computational hardness.
2. If one-way state generators exist, then $\mathsf{BQP} \neq \mathsf{PP}$.

These main results bring along a number of interesting implications. The following are of particular note:

1. A fixed-copy version of one-way state generators exists unconditionally.
2. If we can show that quantum bit commitments exist relative to a PP oracle, then there is a black-box separation showing it is unlikely that we will be able to build one-way state generators from quantum bit commitments.

Preliminaries. We now recall a few key concepts from quantum cryptography before we give more details about the results we show.

Pseudorandom State Generators (PRS). A pseudorandom state generator, originally defined in [JLS18], is a quantum variant of a pseudorandom generator. Given a classical key k, a PRS is a quantum circuit mapping k to a quantum pure state $|\phi_k\rangle$. The security guarantee is that the output of a PRS on a random input should look like a random state. That is, it is hard for any quantum adversary to distinguish any polynomial number of copies of a random $|\phi_k\rangle$ from polynomial copies of a Haar random state.

The relationship between the length of the input key n and the number of output qubits m determines whether a PRS can exist information-theoretically or requires computational assumptions. In particular, [AGQY22] shows that PRSs with output state length $m \geqslant \log n$ qubits can be broken by an inefficient adversary, and thus must be a computational object. On the other hand, it is known that PRSs with state length $m \leqslant c \log n$ exist unconditionally for some $c \in (0, 1)$ [AGQY22, BS20].

One Way State Generators (OWSG). A one-way state generator, originally defined in [MY22a], is a quantum variant of a one-way function. Just like PRSs, a OWSG maps a classical key k to a quantum state $|\phi_k\rangle$. The security guarantee of a OWSG

is that, given any polynomial number of copies of $|\phi_k\rangle$, it is hard for a quantum algorithm to find keys k' such that $|\phi_k\rangle, |\phi_{k'}\rangle$ have noticeable overlap. OWSGs can also be defined to have mixed state outputs [MY22b], although we will not consider this variant in this thesis.

1.4.1 Building one-way state generators from pseudorandom states

It is known that any expanding PRS is also a OWSG [MY22a]. Here, an expanding PRS is one which has keys of length n and output states of length $m > (1+c)n$ for some $c > 0$. We extend this proof to show that any PRS with output length at least $m \geqslant \log n + 1$ implies OWSGs. Since OWSGs require computational hardness [KT23, LC97], and there exists $d < 1$ such that PRSs with output length $\leqslant d \log n$ exist unconditionally [BCQ22], this reduction is close to optimal.

Theorem 1.4.1 (Informal version of Theorem 5.2.3)**.** *For every $c \geqslant 1$, if there exists a PRS mapping n-bit strings to $(\log n + c)$-qubit states, then OWSGs exist.*

Through a closely related argument, we also find that PRSs that map n bits to $\omega(\log n)$ qubits are OWSGs.

Theorem 1.4.2 (Informal version of Theorem 5.2.4)**.** *Any PRS that maps n-bit strings to $\omega(\log n)$-qubit states is also a OWSG.*

1.4.2 Fixed-copy one-way state generators

Both PRSs and OWSGs are defined to be secure against adversaries that are given any polynomial number of copies of the output state. However, we could instead consider an alternative definition where we fix the number of copies given to the adversary. We will refer to these primitives by the names t-copy PRS and t-copy OWSG. Related primitives have already been considered in a number of works, including [GC01, KT23, LMW23]. A summary of prior work and our results can be found in Table 1.1.

It is known that for any fixed function t, expanding $t(\lambda)$-copy PRSs require computational hardness because they can be used to construct quantum bit commitments [LMW23, Yan22, MY22a, BCQ22], where λ is the security parameter. Therefore, any expanding $t(\lambda)$-copy PRS can be broken by an inefficient attacker. On the other hand, if we do not have an expansion requirement, it can be shown that something called an efficient approximate t-design (defined formally in Section 5.1.5) is also a t-copy PRS [Kre21]. Since efficient approximate t-designs exist unconditionally [DCEL09, HMMH+23, OSP23], so do t-copy (non-expanding) PRSs.

Thus, one may ask the question: for what parameters do t-copy OWSGs require computational assumptions? In a recent work, Khurana and Tomer [KT23] show that Wiesner encodings / BB84 states [Wie83, BB14] are 1-copy OWSGs. Additionally, written twenty years before OWSGs were defined, [GC01] shows that t-qubit stabiliser states are $t/2$-copy OWSGs. The OWSG construction of [GC01] only has a weaker security guarantee, but this can be resolved by amplification.

Note that this means that for any fixed polynomial t, $t(\lambda)$-copy OWSGs exist unconditionally, where λ is the security parameter. However, [KT23] shows that quantum bit commitments can be built from $\Theta(n)$-copy OWSGs, where n is the input key length. This is not a contradiction, since the number of copies of security here depends on the input length instead of the security parameter. Thus, we may consider the following refinement of our question:

> *For what functions $t(\cdot)$ do $t(n)$-copy OWSGs require computational assumptions?*

Our proof of Theorem 1.4.2 will also imply the following result.

Corollary 1.4.3 (Informal version of Corollary 5.2.6)**.** *Every efficient approximate t-design mapping n bits to $\omega(\log n)$ qubits is also a $(t-1)$-copy OWSG.*

Primitive	Copies	Security	Comments
Expanding PRS	$t \geqslant 1$ copy	Computational	[MY22a, Yan22, BCQ22]
PRS	$\mathsf{poly}(\lambda)$-copy	Statistical	Approximate t-designs
OWSG	$\mathsf{poly}(\lambda)$-copy	Statistical	Theorem 1.4.2 with approximate t-designs
PRS	$O(n/\log n)$-copy	Statistical	Approximate t-designs
OWSG	$O(\sqrt{n})$-copy	Statistical	Stabiliser states [GC01]
OWSG	$\Omega(n)$-copy	Computational	Implies quantum bit commitments [KT23]
OWSG	$o(n/\log n)$-copy	Statistical	Corollary 1.4.4

Table 1.1: A summary of what is known about the computational and information-theoretic nature of quantum cryptographic primitives, based on the number of copies of the output given to the adversary. We say that security is computational if the existence of the primitive requires computational hardness, and we say that the security is statistical if the primitive can be shown to exist unconditionally against statistical adversaries (in other words, no algorithm, even an inefficient one, is capable of breaking the primitive). The rows above the dashed line correspond to constructions where the number of copies is in terms of the security parameter λ. The rows below the dashed line correspond to constructions where the number of copies is in terms of n, the number of input bits.

If we consider state-of-the-art constructions of approximate t-designs [OSP23], we in addition prove the following.

Corollary 1.4.4 (Equivalent to Corollary 5.2.7)**.** *For every $t(n) = o(n/\log n)$, there exists a $t(n)$-copy OWSG.*

Aside from demonstrating an interesting new property of approximate t-designs, these results give an interesting dichotomy: OWSGs require computational hardness for $\Omega(n)$-copies, and exist unconditionally for $o(n/\log n)$ copies.

1.4.3 A quantum lower bound for PP from a cryptographic assumption

It is known from [Kre21, AGQY22] that the existence of a PRS which outputs a $(\log n + O(1))$-qubit state implies that $\mathsf{BQP} \neq \mathsf{PP}$, where BQP refers to the class of problems efficiently solvable by quantum computers, and PP refers to the class of problems such that a probabilistic Turing machine gets the correct answer with probability strictly greater than $\frac{1}{2}$. The complexity classes BQP and PP satisfy $\mathsf{BQP} \subseteq \mathsf{PP}$ [ADH97] (see Figure 1.1), and it is not yet known whether they are equal or not. Thus, like with one-way functions, the existence of PRSs would bring new implications in complexity theory.

However, no similar results are known about OWSGs or quantum bit commitments. In fact, it is conjectured by [LMW23] that quantum bit commitments may exist relative to a random oracle and *any* classical oracle, even ones that depend on the random oracle. Khurana and Tomer [KT23] observe that there exists a classical oracle that breaks OWSGs, which implies that such a conjecture cannot extend to the existence of OWSGs. If the conjecture of [LMW23] is proven, this would provide a black box separation between OWSGs and quantum bit commitments. However, it is not immediately clear that the oracle [KT23] mentions lies inside any interesting complexity class. Furthermore, the question about whether a PP oracle can break OWSGs was asked in [MY22a].

We show that the existence of OWSGs indeed does have interesting complexity implications. In particular, we show the following.

Theorem 1.4.5 (Informal version of Corollary 5.3.5)**.** *If OWSGs exist, then $\mathsf{BQP} \neq \mathsf{PP}$.*

It then follows that a black box separation between quantum bit commitments and OWSGs can be achieved by proving a weaker version of the conjecture of [LMW23], namely that there exists an oracle $\mathcal{O}$ relative to which quantum bit commitments exist and $\mathsf{PP}^{\mathcal{O}} \subseteq \mathsf{BQP}^{\mathcal{O}}$.

1.4.4 Concurrent and further work

We remark that a concurrent revision of [MY22a] provides a different proof of Theorem 1.4.1. In particular, in their Appendix C they show that, given a PRS $G : k \mapsto |\phi_k\rangle$ mapping n bits to $m \geqslant \log n$ qubits, the mapping $k \mapsto |\phi_k\rangle^{\otimes n}$ is a OWSG. Note, however, that these techniques do not easily extend to imply our other results. In particular, our results show how to build compressing OWSGs (as in Theorem 1.4.2) and thus also give better parameters for unconditional t-copy OWSGs (as in Corollary 1.4.3).

After our results were announced, Hhan, Morimae and Yamakawa [HMY23] proved that there does not exist OWSGs with $O(\log n)$-output qubits. This shows that our proof that PRSs with $\omega(\log n)$-output qubits *are* OWSGs (Theorem 1.4.2) is optimal under the cryptographic assumption that (post-quantum) one-way functions exist, since one-way functions imply PRSs of arbitary output length [BS20].

Chapter 2

Technical overview

This chapter will give an overview of the technical ingredients and insights that are employed in the following chapters (Chapters 3 to 5).

First, a few more general introductory remarks. Chapter 3 applies the technique of random restrictions, first considered by Subbotovskaya [Sub61] and now widely applied in circuit complexity. Random restrictions have been very fruitful in the study of weaker circuit classes, such as AC^0 circuits [Hås86, IMP12], De Morgan formulas [KR13] and branching programs [IMZ19], both for the proof of lower bounds and the construction of algorithms [CKK+15]. However, as observed by Gál and Robere [GR20], there are technical challenges when trying to apply this approach to comparator circuits. In Chapter 3, we successfully apply the method of random restrictions to comparator circuits for the first time.

In Chapter 4, instead of applying direct combinatorial methods to analyse a circuit model as in the previous chapter, our arguments will combine in novel ways several previously unrelated ideas from the literature, such as monotone circuit lower bounds, depth-reduction techniques, and the algebraic framework of *polymorphisms*, standard in the study of CSPs [BCRV03, BCRV04]. The latter will be employed to design reductions between different monotone problems.

In our final technical chapter (Chapter 5), we will closely study and apply the *concentration of Haar states* phenomenom, together with state-of-art t-designs [OSP23], in order to obtain nearly optimal reductions from PRSs to OWGSs, and constructions of unconditional fixed-copy OWSGs. Moreover, we will construct a language in PP with which we can, in polynomial time, estimate the conditional probability of a sampler. Using a recent reduction from one-way state generators to *one-way puzzles* [KT23], this will be enough to break OWSGs. We view our strategy as inspired by Kretschmer's [Kre21] idea for breaking pseudorandom state

generators with a PP oracle, together with a search-to-decision reduction for PP. Contrapositively, this means that the existence of OWSGs imply a (uniform) quantum lower bound for PP.

2.1 Shrinkage for comparator circuits

We will first describe how the method of random restrictions is applied to obtain an optimal average-case lower bound for comparator circuits. The algorithms will then build on the insights obtained in the proof of the lower bound.

Average-case lower bounds. At a high level, the proof of our average-case lower bound is based on the approach developed in [KR13, CKK+15], which can be used to obtain average-case-lower bounds against circuits that admit a property called "shrinkage with high probability under random restrictions". Roughly speaking, this property says that, if we randomly fix the values of some variables in the circuit except for a $0 < p < 1$ fraction of them, then its size shrinks by a factor of p^{Γ} for some $\Gamma > 0$, with very high probability. This method has been used to obtain strong average-case lower bounds against $n^{2.5-o(1)}$-size De Morgan formulas [KR13, CKK+15] (later improved to $n^{3-o(1)}$ by [KRT17]) and $n^{2-o(1)}$-size general formulas and deterministic branching programs.

An obvious issue of applying this approach to comparator circuits is that we don't know how to shrink the size (i.e., number of gates) of a comparator circuit using random restrictions, as when we fix the value of a (non-trivial[1]) wire, we may only be able to remove one gate in the worst scenario (i.e., the gate that is directly connected to that wire). The idea is that instead of shrinking the *number of gates*, we will try to shrink the *number of wires*. The reason why this can be useful is that *one can effectively bound the number of gates of a comparator circuit by its number of wires*; this is a structural result of comparator circuits proved by Gál and Robere [GR20] and was the key ingredient in proving their worst-case lower bound. More precisely, they showed that any comparator circuit that has at most ℓ wires needs no more than ℓ^2 gates (see Lemma 3.1.4). Now following [KR13, CKK+15], one can show that under some certain type of random restriction that leaves a $p := k/n$ fraction of the variables unfixed, for any large enough k, the number of wires of a comparator circuit will shrink (with very high probability) by roughly a factor of p, and hence its number of gates is bounded by $(p \cdot \ell)^2$. By letting $\ell = o\big(n^{1.5}/\big(k \cdot \sqrt{\log n}\big)\big)$, this size is less than $o(n/\log n)$ and from there one can show that the original circuit cannot compute some hard function on more than

[1] We say that a wire is non-trivial if it is connected to some gate.

$1/2 + 1/2^{k^{\Omega(1)}}$ fraction of the inputs.

While the above gives an average-case lower bound, it does not match the worst-case one, because we need to set $k \geqslant \log^c n$ for some (unspecified) constant $c > 1$, which is controlled by the type of random restrictions and the extractor used in the construction of the hard function in both [KR13, CKK+15]. This means we can only achieve a lower bound that is at best $n^{1.5}/(\log n)^{c+.5}$ (even for worst-case hardness). In order to be able to set $k = O(\log n)$, one way is to use a more sophisticated (so called non-explicit bit-fixing) extractor shown in [KRT17], which will allow us to set $k \in \left[O(\log n), \Omega\left(n^{1/3}\right)\right]$ (with hardness $1/2 + 1/2^{\Omega(k)}$). Here we refine and simplify this approach in the case of comparator circuits by using a more structural (block-wise) random restriction that shrinks the number of wires with probability one. Such a random restriction, when combined with a simple extractor, allows us to set $k \in [O(\log n), \Omega(n)]$.

#SAT algorithms. Based on the above analysis in showing average-case lower bounds, one can try to design a SAT algorithm for comparator circuits in a way that is similar to that of [CKK+15], which combines "shrinkage under restrictions" with a memorisation technique. Suppose we have a comparator circuit C with $s := o\left(n^{1.5}/(k \cdot \sqrt{\log n})\right)$ gates and and $\ell \leqslant s$ non-trivial wires. By partitioning the variables into n/k equal-size blocks, we can show that there is some block S_i such that after fixing the variables outside of this block, the number of wires in the restricted circuit is at most $\ell_0 := \ell/(n/k) \leqslant o\left(\sqrt{n/\log n}\right)$. Again by the structural property of comparator circuits (Lemma 3.1.4), this restricted circuit, which is on k variables, has an equivalent circuit with $o(n/\log n)$ gates. Then to count the number of satisfying assignments for the original circuit, we can first memorise the numbers of satisfying assignments for all circuits with at most with $o(n/\log n)$ gates. There are $2^{o(n)}$ of them and hence we can compute in time $2^k \cdot 2^{o(n)}$ a table that stores those numbers. We then enumerate all possible 2^{n-k} restrictions $\rho \in \{0,1\}^{[n]\setminus S_i}$ and for each ρ we look up the number of satisfying assignments of the restricted circuit $C\restriction_\rho$ from the pre-computed table. Summing these numbers over all the ρ's gives the number of satisfying assignments of C.

However, there is a subtle issue in the above argument: although we know that a restricted circuit has an equivalent simple circuit with $o(n/\log n)$ gates, we do not know which simple circuit it is equal to. Note that when we fix the value of a (non-trivial) wire, we may only be able to remove one gate, so the number of gates left in the restricted circuit is possibly $s - (\ell - \ell_0)$, which can be much larger than $n/\log n$, and it is not clear how we can further simplify such a circuit *efficiently*. To overcome this issue, we explore structural properties of comparator circuits to show

how to construct a more sophisticated data structure that not only can tell us the number of satisfying assignments of a circuit with $o(n/\log n)$ gates but also allows us to *efficiently* simplify each restricted circuit to an equivalent circuit with at most this many gates.

Pseudorandom generators and MCSP lower bounds. Our PRG against comparator circuits builds upon the paradigm of [IMZ19], which was used to construct PRGs against circuits that admit "shrinkage under *pseudorandom* restrictions". As in the proof of our average-case-lower bound, in order to apply this paradigm, we will shrink the number of wires instead of the number of gates. Following [IMZ19], we prove a pseudorandom shrinkage lemma for comparator circuits, which can then be used to obtain a PRG of seed length $s^{2/3+o(1)}$, where s is the size of a comparator circuit.

As observed in [CKLM20], one can modify the construction of the PRG in [IMZ19] to make it "locally explicit". This means that, for every seed, the output of the PRG, when viewed as a truth table of a function, has circuit complexity that is about the same as the seed length. Such a "local" PRG immediately implies that MCSP cannot be computed by comparator circuits of size $n^{1.5-o(1)}$, when the size parameter of MCSP is nearly-maximum (i.e., $n/O(\log n)$) [2]. Furthermore, we show a better trade-off between the size parameters of MCSP and the lower bound size of the comparator circuits, as in Theorem 1.2.4. This is similar to what was done by [CJW20] in the case of MCSP lower bounds against De Morgan formulas.

2.2 Constant-depth circuits vs. monotone circuits

The exposition below follows the order in which the results appear above, except for the overview of the proof of Theorem 1.3.1, which appears last. We discuss this result after explaining the proof of Theorem 1.3.2 and the classification of the monotone complexity of CSPs (Theorem 1.3.4 and Theorem 1.3.5), as this sheds light into how the proof of Theorem 1.3.1 was discovered and into the nature of the argument.

A monotone circuit size lower bound for a function in $\mathsf{AC}^0[\oplus]$. We first give an overview of the proof of Theorem 1.3.2.

[2] Note that MCSP takes two input parameters: a truth table and a size parameter θ, and asks whether the given truth table has circuit complexity at most θ.

The lower bound of [GKRS19]. We begin by providing more details about the monotone circuit lower bound of [GKRS19], since their result is a key ingredient in our separation (see [dRGR22] for a more detailed overview). Recall that their function f^{GKRS} corresponds to $\mathsf{CSP\text{-}SAT}_S$ for $S = \{\oplus_3^0, \oplus_3^1\}$. Following their notation, this is simply the Boolean function $\mathsf{3\text{-}XOR\text{-}SAT}_n \colon \{0,1\}^{2n^3} \to \{0,1\}$ which uses each input bit to indicate the presence of a linear equation with exactly 3 variables. This (monotone) function accepts a given linear system over $\mathbb{F}_2$ if the system is *unsatisfiable*. As one of their main results, [GKRS19] employed a lifting technique from communication complexity to show the existence of a constant $\varepsilon > 0$ such that $\mathsf{mSIZE}(\mathsf{3\text{-}XOR\text{-}SAT}_n) = 2^{n^\varepsilon}$. (We show in Appendix B.1 that a weaker super-polynomial monotone circuit size lower bound for $\mathsf{3\text{-}XOR\text{-}SAT}_n$ can also be obtained using the approximation method and a reduction.)

Sketch of the proof of Theorem 1.3.2. Since $\mathsf{3\text{-}XOR\text{-}SAT}_n \in \mathsf{NC}^2$ (see, e.g, [GKRS19]), their result implies that $\mathsf{NC}^2 \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{n^{\Omega(1)}}]$. On the other hand, we are after a separation between *constant-depth* (non-monotone) circuits and *polynomial-size* (unbounded depth) monotone circuits. There are two natural ways that one might try to approach this challenge, as discussed next.

First, the lifting framework explored by [GKRS19] offers in principle the possibility that by carefully picking a different set S of Boolean relations, one might be able to reduce the non-monotone depth complexity of $\mathsf{CSP\text{-}SAT}_S$ while retaining super-polynomial monotone hardness. However, Theorem 1.3.4 shows that this is impossible, as explained above.

A second possibility is to combine the *exponential* 2^{n^ε} monotone circuit size lower bound for $\mathsf{3\text{-}XOR\text{-}SAT}_n$ and a padding argument, since we only need *super-polynomial* hardness. Indeed, this argument can be used to define a monotone function $g \colon \{0,1\}^n \to \{0,1\}$ that is computed by polynomial-size fan-in two circuits of depth $\mathsf{poly}(\log \log n)$ but requires monotone circuit of size $n^{\omega(1)}$. However, it is clear that no padding argument alone can reduce the non-monotone circuit depth bound to $O(1)$ while retaining the desired monotone hardness.

Given that both the classical widely investigated approximation method for monotone lower bounds and the more recent lifting technique do not appear to work in their current forms, for some time it seemed to us that, if true, a significantly new technique would be needed to establish a separation similar to the one in Theorem 1.3.2.

Perhaps surprisingly, it turns out that a more clever approach that combines padding with a non-trivial circuit upper bound can be used to obtain the result.

The first key observation, already present in [GKRS19] and other papers, is that 3-XOR-SAT_n can be computed not only in NC^2 but actually by polynomial-size span programs over $\mathbb{F}_2$. On the other hand, it is known that this model is equivalent in power to parity branching programs [KW93], which correspond to the non-uniform version of $\oplus\mathsf{L}$, i.e., counting modulo 2 the number of accepting paths of a nondeterministic Turing machine that uses $O(\log n)$ space. A second key idea is that such a computation can be simulated by $\mathsf{AC}^0[\oplus]$ circuits of sub-exponential size and large depth. More precisely, similarly to an existing simulation of NL (nondeterministic logspace) by AC^0 circuits of depth d and size $2^{n^{O(1/d)}}$ via a "guess-and-verify" approach, it is possible to achieve an analogous simulation of $\oplus\mathsf{L}$ using $\mathsf{AC}^0[\oplus]$ circuits (this folklore result appears implicit in [AKR+01] and [OSS19]). Putting everything together, it follows that for a large enough but constant depth, 3-XOR-SAT_n can be computed by $\mathsf{AC}^0[\oplus]$ circuits of size $2^{n^{\varepsilon/2}}$. Since this function is hard against monotone circuits of size $2^{n^{\varepsilon}}$, a padding argument can now be used to establish a separation between $\mathsf{AC}^0[\oplus]$ and $\mathsf{mSIZE}[\mathsf{poly}]$. (A careful choice of parameters provides the slightly stronger statement in Theorem 1.3.2.)

Non-trivial monotone simulations and their consequences. In order to conclude that significantly stronger monotone simulations imply new complexity separations (Theorem 1.3.3), we argue contrapositively. By supposing a complexity collapse, we can exploit known monotone circuit lower bounds to conclude that a hard monotone function exists in a lower complexity class. For instance, if $\mathsf{NC}^2 \subseteq \mathsf{NC}^1$, then $\text{3-XOR-SAT} \in \mathsf{NC}^1$, and we can conclude by standard depth-reduction for NC^1 and padding, together with the exponential lower bound for 3-XOR-SAT due to [GKRS19], that there exists a monotone function in AC^0 which is hard for polynomial-size monotone circuits. The other implications are argued in a similar fashion. In particular, we avoid the more complicated use of hardness magnification from [CHO+20] to establish this kind of result, while also getting a stronger consequence.

A little more work is required in the case of graph properties (Theorem 1.3.3 Items 5 and 6), as padding the function computing a graph property does not yield a graph property. We give a general lemma that allows us to pad monotone graph properties while preserving their structure (Lemma 4.2.7). We then argue as in the case for general functions, using known monotone lower bounds for graph properties. We note that Lemma 4.2.7 is also important in the proof of Theorem 1.3.1, which will be discussed below. We believe that our padding technique for graph properties might find additional applications.

Monotone complexity of CSPs. These are the most technical results of Chapter 4. Since explaining the corresponding proofs requires more background and case analysis, here we only briefly describe the main ideas and references behind Theorem 1.3.4, Theorem 1.3.5, and the extensions discussed in Section 4.4.

A seminal work of Schaefer [Sch78] proved that any Boolean CSP is either solvable in polynomial-time or it is NP-complete. Later, Jeavons [Jea98] observed that the complexity of deciding if a given set of constraint applications of S is satisfiable depends exclusively on the set $\mathsf{Pol}(S)$ of *polymorphisms* of S. Intuitively, the set of polymorphisms of a set of relations is a measure of its symmetry. The more symmetric a set of relations is, the lesser is its expressive power. Jeavons formally proves this intuition by showing that, if $\mathsf{Pol}(S) \subseteq \mathsf{Pol}(S')$, then the problem of deciding the satisfiability of a given S'-formula can be reduced in polynomial-time to that of deciding the satisfiability of a given S-formula. This allows Jeavons to reprove Schaefer's result.

Existing proofs and classification results for constraint satisfaction problems do not encode the satisfiability problem as a monotone Boolean function $\mathsf{CSP\text{-}SAT}_S$, in the way we described above. We reexamine Schaefer's and Jeavons's proofs and establish that the reduction from $\mathsf{CSP\text{-}SAT}_{S'}$ to $\mathsf{CSP\text{-}SAT}_S$ can also be done with efficient monotone circuits. Making use of and adapting parts of the refined results and analysis of [ABI+09], which builds on the earlier dichotomy result of [Sch78] and provides a detailed picture of the computational complexity of Boolean-valued CSPs, we prove in fact that the underlying reductions can all be done in monotone nondeterministic logspace.

Finally, using known upper and lower bounds for monotone circuits together with a direct analysis of some basic cases, and inspecting Post's lattice [Pos41, BCRV03, BCRV04], we are able to show that $\mathsf{CSP\text{-}SAT}_S$ is hard for monotone circuits only when $\mathsf{CSP\text{-}SAT}_S$ is $\oplus$L-complete, as in Theorem 1.3.4 Part 1.

A monotone circuit depth lower bound for a function in AC^0. Next, we combine insights obtained from the monotone lower bound of [GKRS19], our proof of Theorem 1.3.2 via a guess-and-verify depth reduction and padding, and the statement of Theorem 1.3.4 (limits of the direct approach via CSPs) to get the separation in Theorem 1.3.1. As alluded to above, our approach differs from those of [Oko82, AG87, BST13, COS17] and related results in the context of first-order logic [Sto95, Kup21, Kup22].

Recall that the [GKRS19] framework lifts a Resolution width lower bound for an unsatisfiable S-formula F into a corresponding monotone circuit size lower bound

for $\mathsf{CSP\text{-}SAT}_S$. On the other hand, Theorem 1.3.4 rules out separating constant-depth circuits from monotone circuits of polynomial size via $\mathsf{CSP\text{-}SAT}_S$ functions. In particular, we cannot directly apply the chain of reductions from [GKRS19] to obtain the desired separation result. Instead, we extract from the specific S-formula F that they use a *structural property* that will allow us to improve the $\mathsf{AC}^0[\oplus]$ upper from Theorem 1.3.2 to the desired AC^0 upper bound in Theorem 1.3.1.

In [GKRS19] the formula F is a *Tseitin* contradiction, a well-known class of unsatisfiable CNFs with a number of applications in proof complexity. For an undirected graph G, the Tseitin formula $T(G)$ encodes a system of linear equations modulo 2 as follows: each edge $e \in E(G)$ becomes a Boolean variable x_e, and each vertex $v \in V(G)$ corresponds to a constraint (linear equation) C_v stating that $\sum_{u \in N_G(v)} x_{\{v,u\}} = 1 \pmod 2$, where $N_G(v)$ denotes the set of neighbours of v in G. Crucially, $T(G)$ does not encode an arbitrary system of linear equations, i.e., the following key structural property holds: every variable x_e appears in exactly 2 equations.

On a technical level, this property is not preserved when obtaining a (total) monotone function $\mathsf{CSP\text{-}SAT}_S$ by the gadget composition employed in the lifting framework and its reductions. However, we can still hope to explore this property in a somewhat different argument with the goal of obtaining CSP instances that lie in a complexity class weaker than $\oplus\mathsf{L}$, which is the main bottleneck in the proof of Theorem 1.3.2 yielding $\mathsf{AC}^0[\oplus]$ circuits instead of AC^0. At the same time, considering this structural property immediately takes us outside the domain of Theorem 1.3.4, which does not impose structural conditions over the CSP instances.

We can capture the computational problem corresponding to this type of system of linear equations using the following Boolean function. Let $\mathsf{OddFactor}_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be the function that accepts a given graph G if the formula $T(G)$ described above is *satisfiable*. (Equivalently, if G admits a spanning subgraph in which the degree of every vertex is odd.) Note that $\mathsf{OddFactor}_n$ is a monotone Boolean function: adding edges to G cannot make a satisfiable system unsatisfiable, since we can always set a new edge variable x_e to 0.

While 3-XOR-SAT (the corresponding $\mathsf{CSP\text{-}SAT}_S$ function obtained from an appropriate Tseitin formula via the framework of [GKRS19]) admits a $\oplus\mathsf{L}$ upper bound, we observe that $\mathsf{OddFactor}_n$ can be computed in L thanks to its more structured class of input instances. Indeed, one can prove that the formula $T(G)$ is satisfiable if and only if every connected component of G has an even number of vertices.[3] In turn, the latter condition can be checked in logarithmic space using

[3] A simple parity argument shows that odd-sized components cannot be satisfied. On the other

Reingold's algorithm for undirected s-t-connectivity [Rei05]. (We note that related ideas appear in an unpublished note of Johannsen [Joh03].) This is the first application of Reingold's algorithm to this kind of separation.

At the same time, $\mathsf{OddFactor}_n$ retains at least part of the monotone hardness of 3-XOR-SAT. Using a different reduction from a communication complexity lower bound, [BGW99] proved that the monotone circuit depth of $\mathsf{OddFactor}_n$ is $n^{\Omega(1)}$. Altogether, we obtain a monotone Boolean function (indeed a graph property) that lies in L but is not in $\mathsf{mDEPTH}[n^{o(1)}]$. Applying a guess-and-verify depth reduction for L and using (graph) padding (analogously to the proof sketch of Theorem 1.3.2), we get a monotone graph property in AC^0 that is not in $\mathsf{mDEPTH}[\log^k n]$. This completes the sketch of the proof of Theorem 1.3.1.

2.3 Quantum one-wayness

We will now give an overview of the main technical ingredients of the results of Chapter 5.

Building OWSGs from PRSs. We first review our improved construction of one-way state generators from pseudorandom states.

OWSGs from shrinking PRSs. Let us first review the argument from [MY22a] that an expanding PRS is also a OWSG. Recall that an expanding PRS maps n-bit strings to m-qubit quantum states, where $m \geqslant (1+c)n$ for some $c > 0$. The natural reduction which uses the OWSG adversary to also break the PRS works. Let $\mathcal{A}$ be a OWSG adversary which outputs k' such that $|\phi_{k'}\rangle$ is close to $|\phi_k\rangle$ using t copies of $|\phi_k\rangle$. Given $t+1$ copies of a state $|\psi\rangle$, we can test whether it is an output of the PRS or Haar-random as follows: run $\mathcal{A}$ on the first t copies to get a state $|\phi_{k'}\rangle$, and compare $|\phi_{k'}\rangle$ with the last copy of $|\psi\rangle$. If $|\psi\rangle = |\phi_k\rangle$ for some k, then $|\phi_{k'}\rangle$ will be close to $|\phi_k\rangle$. If $|\psi\rangle$ is Haar-random, then since it is a random state on $(1+c)n$ qubits, with high probability it is far from $|\phi_k\rangle$ for all 2^n values of k. This is because $\frac{2^n}{2^{(1+c)n}}$ is negligible in n.

To improve this result to PRSs with $O(\log n)$-bit output, we simply improve the analysis of the exact same reduction. We make the following simple observation about Haar-random states: for any fixed m-qubit state $|\phi\rangle$, the probability that a

hand, we can always satisfy an even-sized component by starting with an arbitrary assignment, which must satisfy an even number of constraints by a parity argument, and flipping the values of the edges in a path between unsatisfied nodes, until all nodes in the connected component are satisfied.

Haar-random state $|\psi\rangle$ is "close" to $|\phi\rangle$ is $2^{-\Omega(2^m)}$. Thus, in the reduction above, if $|\psi\rangle$ is a Haar-random state on $\omega(\log n)$ qubits, then, by a union bound, with high probability it is far from $|\phi_k\rangle$ for all 2^n values of k. This argument is enough to derive Theorem 1.4.2. Using amplification [MY22b], we also find that OWSGs can be built even from a PRS which outputs $\log n + O(1)$ qubits, obtaining Theorem 1.4.1.

Building OWSGs from a fixed-copy PRS. We can instantiate our reduction with a t-copy PRS (i.e., a PRS that is secure against t copies). Our reduction then shows that any t-copy PRS is also a $(t-1)$-copy OWSG. The fact that approximate t-designs are t-copy PRSs [Kre21] then gives Corollary 1.4.3.

Recently, O'Donnell, Servedio, and Paredes [OSP23] showed that there exists an efficient $2^{-\lambda}$-approximate t-design on m-qubit quantum states with seed length $n = O(mt + \lambda)$ (see Section 5.1.5 for a definition of t-designs). Setting $m = \omega(\log n)$ and $t = o\left(\frac{n}{\log n}\right)$ shows that approximate $o\left(\frac{n}{\log n}\right)$-designs with $\omega(\log n)$-output bits exist, and thus $o\left(\frac{n}{\log n}\right)$-copy OWSGs also exist. That is, Corollary 1.4.4 holds.

Breaking one-way state generators with a PP oracle. The complexity class PP is typically defined by referring to Turing machines or randomised algorithms. These definitions are not very useful when dealing with quantum computing. However, it turns out that PP has an equivalent formulation with much more obvious quantum applications, known as PostBQP [Aar05]. PostBQP refers to the class of problems efficiently solvable by uniform quantum circuits with the additional ability to *postselect*. Postselection is another word for performing conditional sampling, and in the quantum setting refers to the ability to choose the result of a measurement and acquire the corresponding residual state. Thus, to break any OWSG with a PP oracle, it suffices to define an algorithm which breaks the OWSG given oracle access to some language in PostBQP.

Instead of breaking OWSGs with a PostBQP oracle directly, we rely on a recent result which constructs an interesting classical output primitive, a one-way puzzle, from any OWSG [KT23]. Formally, a one-way puzzle is a pair of algorithms $(\mathsf{Samp}, \mathsf{Ver})$ where Samp samples a key-puzzle pair (k, s) such that $\mathsf{Ver}(k, s)$ outputs 1 with overwhelming probability. Samp is required to be an efficient quantum algorithm, and Ver is allowed to be any arbitrary function. The security requirement is that given s, it is hard for an adversary to find a k' such that $\mathsf{Ver}(k', s) = 1$. Morally, a one-way puzzle is a one-way function where the input and output are sampled together.

It is clear that any one-way puzzle $(\mathsf{Samp}, \mathsf{Ver})$ can be broken by an adversary

with the ability to postselect. Given a puzzle s, the adversary can simply run Samp up until it would measure the output state, and then postselect on the output puzzle being s. Measuring the output key will then give a k' which with high probability will satisfy $\mathsf{Ver}(s, k') = 1$. This attack can be viewed as a search version of the attack of [Kre21] against any λ-output PRS.

But note that this attack makes use of postselecting directly. It is unclear how to translate this into an attack with a decision oracle for PP. To solve this, we show that it is possible in general to perform conditional sampling given access to a PP oracle:

Lemma 2.3.1 (Informal version of Lemma 5.3.2)**.** *Let* Samp *be a (uniform) quantum polynomial time algorithm such that* $\mathsf{Samp}(1^n)$ *outputs a pair of classical strings* (k, s)*. There exists a poly-time quantum algorithm* $\mathcal{A}$ *and a* PP *language* $\mathcal{L}$ *such that* $\mathcal{A}^{\mathcal{L}}$ *takes as input* s' *and outputs* k'*, and whose distribution has total variation distance at most* $1/n$ *from the distribution* $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$ *defined by*

$$\mathbb{P}[(\mathsf{Samp} \,|\, s')^{\mathsf{key}} \to k'] = \mathop{\mathbb{P}}_{\mathsf{Samp}(1^n) \to (k,s)}[k = k' | s = s'].$$

(In other words, we denote by $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$ *the distribution of keys generated by* $\mathsf{Samp}(1^n)$ *conditioned on the puzzle being equal to* s'*.)*

This lemma is in some sense a "search-to-decision" style argument for PP. The argument goes along the same lines as the search-to-decision reduction for SAT. Note that with a $\mathsf{PostBQP}$ oracle, we can test whether or not it is possible for an algorithm Samp to produce any given output x by simply postselecting on x being produced by Samp.

A naive approach to sampling k' from $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$ is to sample k' bit by bit. It is possible with a PP oracle to check whether any given output is in the range of Samp. Thus, we can begin by checking whether $(1, s')$ is in the range of Samp with all but the first bit of the key discarded. If so, we can set the first bit of k' to be 1, otherwise 0. In the next step, we can check whether $(k'_1 \circ 1, s')$ is in the range of Samp with all but the second bit of the key discarded. If so, we can set the second bit of k' to be 1, otherwise 0. Repeating this process for each bit of the key will uniformly select an output k' from the range of $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$.

However, $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$ is not necessarily a flat distribution, and so the resulting distribution on k' may be very different from $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$. However, this can be resolved by noticing a key fact. Using a PP oracle, it is possible to *estimate the probability* that the first bit of the output of Samp is 1 conditioned on the output puzzle being s'. Thus, we can use the same technique as before, but instead of just

setting each bit of k', we can sample each bit according to our approximation of the correct conditional distribution. Although the error will add up, we can set our initial error to be small enough that the distribution over k' will be sufficiently close to $(\mathsf{Samp}\,|\,s')^{\mathsf{key}}$.

Applying Lemma 2.3.1 to our postselecting attack against one-way puzzles gives us an efficient quantum attack against one-way puzzles using a PP oracle. As one-way puzzles can be built from one-way state generators, this immediately implies Theorem 1.4.5.

More on search-to-decision reductions using a PP *oracle.* The ability of a postselection oracle to aid in search-to-decision reductions was first noted by [INN+22], where it was shown that a quantum poly-time algorithm can find a QMA witness by making one quantum query to a PP oracle. They use very different techniques, and in fact our algorithm requires many classical queries instead of one quantum query.

Chapter 3

Algorithms and Lower Bounds for Comparator Circuits from Shrinkage

Abstract

Comparator circuits are a natural circuit model for studying bounded fan-out computation whose power sits between nondeterministic branching programs and general circuits. Despite having been studied for nearly three decades, the first superlinear lower bound against comparator circuits was proved only recently by Gál and Robere [GR20], who established a $\Omega\big((n/\log n)^{1.5}\big)$ lower bound on the size of comparator circuits computing an explicit function of n bits.

In this chapter, we initiate the study of *average-case complexity* and *circuit analysis algorithms* for comparator circuits. Departing from previous approaches, we exploit the technique of shrinkage under random restrictions to obtain a variety of new results for this model. Among them, we show

– **Average-case Lower Bounds (Section 3.2).** For every $k = k(n)$ with $k \geqslant \log n$, there exists a polynomial-time computable function f_k on n bits such that, for every comparator circuit C with at most $n^{1.5}/O\big(k \cdot \sqrt{\log n}\big)$ gates, we have

$$\mathop{\mathbb{P}}_{x \in \{0,1\}^n} [C(x) = f_k(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}}.$$

This average-case lower bound matches the worst-case lower bound of [GR20] by letting $k = O(\log n)$.

– **#SAT Algorithms (Section 3.4).** There is an algorithm that counts the number of satisfying assignments of a given comparator circuit with at most $n^{1.5}/O\big(k \cdot \sqrt{\log n}\big)$

gates, in time $2^{n-k} \cdot \mathsf{poly}(n)$, for any $k \leqslant n/4$. The running time is non-trivial (i.e., $2^n/n^{\omega(1)}$) when $k = \omega(\log n)$.

– **Pseudorandom Generators and MCSP Lower Bounds (Section 3.5).** There is a pseudorandom generator of seed length $s^{2/3+o(1)}$ that fools comparator circuits with s gates. Also, using this PRG, we obtain an $n^{1.5-o(1)}$ lower bound for MCSP against comparator circuits.

Organisation of the chapter

After reviewing preliminary definitions and results in Section 3.1, we will present our average-case lower bounds against comparator circuits in Section 3.2, and average-case lower bounds against various other circuit models in Section 3.3. We will then present and prove our #SAT algorithms (Section 3.4), pseudorandom generators and a MCSP lower bound (Section 3.5), and a learning algorithm (Section 3.6).

3.1 Preliminaries

In this section, we review some basic definitions, including that of random restrictions and comparator circuits, which will be used throughout the chapter. We also review structural properties of comparator circuits based on previous work.

3.1.1 Definitions and notations

For $n \in \mathbb{N}$, we denote $\{1, \ldots, n\}$ by $[n]$. Fix a universal Turing machine U. For a string x, the *Kolmogorov complexity* of x, denoted as $\mathrm{K}_U(x)$, is the length of the smallest binary string ρ such that $U(\rho) = x$. It's known that, for any sufficiently efficient universal Turing machine, the value of $\mathrm{K}_U(x)$ does not change by more than an additive constant for all x (see, e.g., [LV08, Theorem 2.11]). Since such a difference will not affect any of our results, we will omit U and simply write $\mathrm{K}(x)$.[1]

Restrictions. A *restriction* for an n-variate Boolean function f, denoted by $\rho \in \{0, 1, *\}^n$, specifies a way of fixing the values of some subset of variables for f. That is, if $\rho(i)$ is $*$, we leave the i-th variable unrestricted and otherwise fix its value to be $\rho(i) \in \{0, 1\}$. We denote by $f\restriction_\rho \colon \{0,1\}^{\rho^{-1}(*)} \to \{0,1\}$ the restricted function after the variables are restricted according to ρ, where $\rho^{-1}(*)$ is the set of unrestricted variables.

[1] We also note that variations on the number of tapes of U and similar considerations will also not change the Kolmogorov complexity by more than an additive constant.

Comparator circuits. We define comparator circuits as a set of *wires* labelled by an input literal (a variable x_i or its negation $\neg x_i$) or a Boolean constant (0 or 1), a sequence of *gates*, which are *ordered* pairs of wires, and a designated *output wire*. In other words, each gate is a pair of wires (w_i, w_j), denoting that the wire w_i receives the logical conjunction ($\wedge$) of the wires, and w_j receives the logical disjunction ($\vee$). On a given input a, a comparator circuit computes as follows: each wire labelled with a literal x_i is initialised with a_i, and we update the value of the wires by following the sequence of gates; the output wire contains the result of the computation. A wire is called *non-trivial* if there is a gate connected to this wire. Note that, if a comparator circuit has ℓ non-trivial wires and s gates, then $\ell \leqslant s$. This means that lower bounds on the number of wires also imply lower bounds on the number of gates.

As discussed in the Introduction (Section 1.2), comparator circuits can be seen as a model of circuits of bounded fan-out. In particular, this means that comparator circuits, like formulas, cannot create a COPY gate, which creates two copies of the same input bit. This follows more formally from the definition above by noticing that the output of a comparator circuit always has the same Hamming weight of its input.

3.1.2 Structural properties of comparator circuits

For a gate g in a comparator circuit and an input $x \in \{0,1\}^n$, we denote by $u_g(x)$ (resp. $v_g(x)$) the first (resp. second) in-value to the gate g when given x as input to the circuit.

Definition 3.1.1 (Useless Gates)**.** *We say that a gate g in a comparator circuit is* useless *if either one of the following is true:*

1. *for every input x, $(u_g(x), v_g(x)) \in \{(0,1),(0,0),(1,1)\}$.*
2. *for every input x, $(u_g(x), v_g(x)) \in \{(1,0),(0,0),(1,1)\}$.*

We say that a useless gate is of TYPE-1 *(resp.* TYPE-2*) if it is the first (resp. second) case. Also, a gate is called* useful *if it is not useless.*

The following proposition allows us to remove useless gates from a comparator circuit. It has a simple proof which we give here for completeness.

Proposition 3.1.2 ([GR20, Proof of Proposition 3.2])**.** *Let C be a comparator circuit whose gates are $g_1, g_2, \ldots, g_s$ (where g_s is the output gate) and let $g_i = (\alpha, \beta)$ be any useless gate in C.*

- *Suppose g_i is of TYPE-1. Then the circuit C' obtained from C by removing the gate g_i computes the same function as that of C.*
- *Suppose g_i is of TYPE-2. Let C' be the circuit whose gates are*

$$g_1, g_2, \ldots, g_{i-1}, g'_{i+1} \ldots, g'_s,$$

where for $j = i+1, \ldots, s$, g'_j is obtained from g_j by replacing α with β (if g_j contains α) and at the same time replacing β with α (if g_j contains β). Then C' computes the same function as that of C.

Proof. On the one hand, if g is a TYPE-1 useless gate, then for every input to the circuit, the out-values of g are the same as its in-values, so removing g does not affect the function computed by the original circuit. On the other hand, if g is of TYPE-2, then the in-values feeding to g will get swapped after g is applied. This has the same effect as removing g and "re-wiring" the gates after g so that a gate connecting one of the wires of g gets switched to connect the other wire of g, as described in the second item of the proposition. □

We need the following powerful structural result for comparator circuits from [GR20].

Theorem 3.1.3 ([GR20, Theorem 1.2])**.** *If C be is a comparator circuit with ℓ wires and s gates such that every gate in C is useful, then $s \leqslant \ell \cdot (\ell - 1)/2$.*

Proposition 3.1.2 and Theorem 3.1.3 together give the following lemma.

Lemma 3.1.4. *Every comparator circuit with $\ell > 0$ wires has an equivalent comparator circuit with ℓ wires and with at most $\ell \cdot (\ell - 1)/2$ gates.*

3.2 Average-case Lower Bounds

In this section, we prove our average-case lower bound against comparator circuits. We first describe the hard function.

3.2.1 The hard function

List-decodable codes. Recall that a (ζ, L)-list-decodable binary code is a function $\mathrm{Enc}\colon \{0,1\}^n \to \{0,1\}^m$ that maps n-bit messages to m-bit codewords so that, for each $y \in \{0,1\}^m$, there are at most L codewords in the range of Enc that have relative hamming distance at most ζ from y. We will use the following list-decodable code.

Theorem 3.2.1 (See e.g., [CKK+15, Proof of Theorem 6.4])**.** *There is a constant $c > 0$ such that for any given $k = k(n) > c \cdot \log n$, there exists a binary code* Enc *mapping n-bit message to a codeword of length 2^k, such that* Enc *is (ζ, L)-list-decodable for $\zeta = 1/2 - O\big(n/2^{k/2}\big)$ and $L \leqslant O\big(2^{k/2}/n\big)$. Furthermore, there is a polynomial-time algorithm for computing the i-th bit of* $\mathrm{Enc}(x)$*, for any inputs $x \in \{0,1\}^n$ and $i \in \big[2^k\big]$.*

Definition 3.2.2 (Generalised Andreev's Function)**.** *Let k be a positive integer. Define $A_k \colon \{0,1\}^{n+n} \to \{0,1\}$ as follow:*

$$A_k(x_1, \ldots, x_n, y_1, \ldots, y_n) := \mathrm{Enc}(x_1, \ldots, x_n)_{\alpha(y_1, \ldots, y_n)},$$

where Enc *is the code from Theorem 3.2.1 that maps n bits to 2^k bits, and $\alpha \colon \{0,1\}^n \to \{0,1\}^k$ is defined as*

$$\alpha(y_1, \ldots, y_n) := \left(\bigoplus_{i=1}^{n/k} y_i, \bigoplus_{i=n/k+1}^{2n/k} y_i, \ldots, \bigoplus_{i=(k-1)n/k+1}^{n} y_i \right).$$

That is, the function α partitions y evenly into k consecutive blocks and outputs the parities of the variables in each block.

Note that the function A_k defined above is polynomial-time computable since we can compute $\alpha(y)$ and $\mathrm{Enc}(x)_i$ for any given i in $\mathsf{poly}(n)$ time.

3.2.2 Proof of the average-case lower bound

We will show a lower bound on the number of wires, which automatically implies a lower bound on the number of gates.

Theorem 3.2.3. *There exist constants $c, d \geqslant 1$ such that the following holds. For any $k \geqslant c \cdot \log n$, there is a polynomial-time computable function f_k such that, for every comparator circuit C whose number of wires is*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}},$$

we have

$$\mathop{\mathbb{P}}_{x \in \{0,1\}^n}[f_k(x) = C(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}}.$$

Proof. Let A_k be the generalised Andreev's function on $2n$ variables. Let C be a comparator circuit on $2n$ variables with $\ell \leqslant n^{1.5}/\big(d \cdot k \cdot \sqrt{\log n}\big)$ wires, where $d \geqslant 1$

is a sufficiently large constant. To avoid some technicalities due to divisibility that can be overcome easily, we assume that n is divisible by k.

We need to upper bound the following probability.

$$\begin{aligned}\mathop{\mathbb{P}}_{x,y\in\{0,1\}^n\times\{0,1\}^n}[A_k(x,y)=C(x,y)] &\leqslant \mathop{\mathbb{P}}_{x,y}[A_k(x,y)=C(x,y)\mid \mathrm{K}(x)\geqslant n/2]\\ &\quad+\mathop{\mathbb{P}}_{x}[\mathrm{K}(x)<n/2]\\ &\leqslant \mathop{\mathbb{P}}_{x,y}[A_k(x,y)=C(x,y)\mid \mathrm{K}(x)\geqslant n/2]+\frac{1}{2^{n/2}}.\end{aligned}$$

Let x be any fixed n-bit string with Kolmogorov complexity at least $n/2$. Let $A'\colon \{0,1\}^n\to\{0,1\}$ be

$$A'(y) := A_k(x,y),$$

and let C' be a comparator circuit on n variables with at most ℓ wires defined as

$$C'(y) := C(x,y).$$

We will show that

$$\mathop{\mathbb{P}}_{y\in\{0,1\}^n}\left[A'(y)=C'(y)\right]\leqslant\frac{1}{2}+\frac{n}{2^{k/4}}.$$

First of all, let us divide the n variables of C' into n/k parts, each of which contains k variables, as follows. We first partition the n variables evenly into k *consecutive* blocks, denoted as $B_1,B_2,\ldots,B_k$. Then we define the i-th part S_i, where $i\in[n/k]$, to be the union of the i-th variables in each of $B_1,B_2,\ldots,B_k$. That is

$$S_i := \bigcup_{j\in[k]}\left\{y\colon y \text{ is the } i\text{-th variables of } B_j\right\}.$$

Now we count the number of wires that are labelled by the variables in each S_i and let

$$w_i := \left|\{u\colon u \text{ is a wire labelled by some } x\in S_i \text{ (or its negation)}\}\right|.$$

We have

$$\sum_{i\in[n/k]} w_i = \ell,$$

which implies that there is a particular $i\in[n/k]$ such that

$$w_i\leqslant\frac{\ell}{n/k}\leqslant\frac{1}{d}\cdot\sqrt{\frac{n}{\log n}}=:\ell_0.$$

Next, we will consider restrictions that fix the values of the variables outside

S_i. Note that if we fix the value of a variable x_i in a comparator circuit, then we can obtain a restricted circuit so that all the wires that are labelled by either x_i or $\neg x_i$ are eliminated, after some appropriate updates on the gates in the circuit. This is not an obvious fact. One way to see this is that once we fix the value of a wire, the gate that directly connects this wire becomes *useless* in the sense of Definition 3.1.1 so it can be removed after some appropriate "re-wirings" of the gates in the circuit as described in Proposition 3.1.2. Then we can keep doing this until no gate is connected to that wire, in which case the wire can be removed from the circuit.

Now we have

$$\mathop{\mathbb{P}}_{y\in\{0,1\}^n}\left[A'(y)=C'(y)\right]=\mathop{\mathbb{P}}_{\rho\in\{0,1\}^{[n]\setminus S_i},z\in\{0,1\}^k}\left[A'\restriction_\rho(z)=C'\restriction_\rho(z)\right].$$

It suffices to upper bound

$$\mathop{\mathbb{P}}_{z\in\{0,1\}^k}\left[A'\restriction_\rho(z)=C'\restriction_\rho(z)\right],$$

for every $\rho\in\{0,1\}^{[n]\setminus S_i}$. For the sake of contradiction, suppose for some ρ, we have

$$\frac{1}{2}+\frac{n}{2^{k/4}}<\mathop{\mathbb{P}}_{z\in\{0,1\}^k}\left[A'\restriction_\rho(z)=C'\restriction_\rho(z)\right]=\mathop{\mathbb{P}}_{z\in\{0,1\}^k}\left[\mathrm{Enc}(x)_\alpha=C'\restriction_\rho(z)\right],\tag{3.1}$$

where $\alpha\in\{0,1\}^k$ is

$$\alpha_j:=\mathsf{Parity}\Big(\rho|_{B_j\setminus S_i}\Big)\oplus z_j,$$

and $\rho|_{B_j\setminus S_i}$ denotes the partial assignment given by ρ but restricted to only variables in the set $B_j\setminus S_i$. Note that α is uniformly distributed for uniformly random z. Therefore, if we have the values of $\mathsf{Parity}\Big(\rho|_{B_j\setminus S_i}\Big)$ for each $j\in[k]$ (k bits in total), and if we know the restricted circuit $C'\restriction_\rho$, then we can compute the codeword $\mathrm{Enc}(x)$ correctly on at least $1/2+n/2^{k/4}$ positions, by evaluating $C'\restriction_\rho(z)$ for every $z\in\{0,1\}^k$. As a result, we can list-decode $\mathrm{Enc}(x)$, and, using additional $k/2$ bits (to specify the index of x in the list), we can recover x exactly. Finally, note that the number of wires in $C'\restriction_\rho$ is at most ℓ_0. Therefore, by Lemma 3.1.4, such a circuit can be described using a string of length at most

$$\begin{aligned}O\big(\ell_0\cdot\log(n)+\ell_0^2\cdot\log(\ell_0)\big)&\leqslant O\big(\ell_0^2\cdot\log n\big)\\&=O\left(\frac{n}{d^2\cdot\log n}\cdot\log n\right)\\&\leqslant n/4,\end{aligned}$$

where the last inequality holds when d is sufficiently large. Therefore, we can recover[2] x using less than

$$n/4 + k + k/2 + O(\log n) < n/2$$

bits. Here we assume $k \leqslant n/8$ since otherwise the theorem can be shown trivially. This contradicts the fact that the Kolmogorov complexity of x is at least $n/2$. □

3.3 Tight Average-case Lower Bounds from a Nečiporuk-Type Property

Here, we describe a generalisation of the average-case lower bound in Section 3.2 to circuit classes whose worst-case lower bounds can be proved via Nečiporuk's method.

Theorem 3.3.1. *There is a constant $c > 1$ such that the following holds. Let $\mathcal{C}$ be a class of Boolean circuits that is closed under restrictions. Suppose that, for any $k \in [c \cdot \log n,\, n/3]$, there exists a partition of the n variables into $m := n/k$ equal-sized blocks $S_1, S_2, \ldots, S_m$ and a collection of k-input-bit functions $\mathcal{H}$ such that*

1. *$|\mathcal{H}| \leqslant 2^{n/2}$, and*
2. *for every $C \in \mathcal{C}_n$ of size $s(n,k)$, we have $\{C\restriction_\rho\}_{\rho \in \{0,1\}^{[n]\setminus S_i}} \subseteq \mathcal{H}$ for some block S_i.*

Then for any $k \in [c \cdot \log n,\, n/6]$, there exists a polynomial-time computable function f_k which satisfies

$$\mathop{\mathbb{P}}_{x \in \{0,1\}^n}[C(x) = f_k(x)] \leqslant \frac{1}{2} + \frac{1}{2^{\Omega(k)}},$$

for every $C \in \mathcal{C}_n$ of size $s(n/2, k)$.

Remark. In the original Nečiporuk's argument for getting worst-case lower bounds, it is only required that, for every $C \in \mathcal{C}$, there is some block such that the number of distinct functions, after fixing the variables outside of the block, is at most $2^{n/2}$, and *this set of functions can be different for different C*. For Theorem 3.3.1, we need

[2] Here, 'recover' means that it's possible to write a program for a fixed universal Turing machine with the above information hard-coded into it which outputs x. The entire description of the program will be the length of the concatenation of the above binary strings, plus an $O(1)$ overhead to account for the algorithm itself. This bounds the Kolmogorov complexity of x.

something stronger which says that it is the *same set of* $2^{n/2}$ *functions for every* C. We remark that, though the weaker condition is sufficient for *worst-case* lower bounds, all applications of Nečiporuk's method known to us also prove the stronger condition, thus yielding *average-case* lower bounds by Theorem 3.3.1.

Theorem 3.3.1 requires a slightly different argument than that of Theorem 3.2.3. Its proof is presented in Section 3.3.1 below.

By combining Theorem 3.3.1 with known structural properties for various models (see e.g., [Juk12]), we get that for the class of circuits $\mathcal{C}$ of size s, where

- $\mathcal{C}$ is the class of general formulas, and $s = n^2/O(k)$, or
- $\mathcal{C}$ is the class of deterministic branching programs or switching networks, and $s = n^2/O(k \cdot \log n)$, or
- $\mathcal{C}$ is the class of nondeterministic branching programs, parity branching programs, or span programs, and $s = n^{1.5}/O(k)$,

there exists a function f_k such that $\mathbb{P}_{x\in\{0,1\}^n}[C(x) = f_k(x)] \leqslant 1/2 + 1/2^{\Omega(k)}$ for every $C \in \mathcal{C}$, which matches the state-of-the-art worst-case lower bounds (up to a multiplicative constant) by letting $k = O(\log n)$.

3.3.1 Proof of Theorem 3.3.1

We need to slightly modify the hard function in Definition 3.2.2 (particularly the function α) to adjust an arbitrary partition as in Theorem 3.3.1. For an integer k and a partition of n variables into n/k equal-sized blocks, denoted by $S := \{S_1, S_2, \ldots, S_{n/k}\}$, define $A_{S,k}\colon \{0,1\}^{n+n} \to \{0,1\}$ as follows:

$$A_{S,k}(x_1, \ldots, x_n, y_1, \ldots, y_n) := \mathrm{Enc}(x_1, \ldots, x_n)_{\alpha(y_1,\ldots,y_n)},$$

where Enc is the code from Theorem 3.2.1 that maps n bits to 2^k bits, and $\alpha\colon \{0,1\}^n \to \{0,1\}^k$ is defined as

$$\alpha(y_1, \ldots, y_n) := \left(\bigoplus_{z\in B_1} z, \bigoplus_{z\in B_2} z, \ldots, \bigoplus_{z\in B_k} z \right),$$

where $B_j := \bigcup_{i\in[n/k]} \{z\colon z \text{ is the } j\text{-th variables of } S_i\}$.

Good x. We will need the following lemma which says that for most $x \in \{0,1\}^n$, the codeword of x is hard to approximate for any fixed small set of functions.

Lemma 3.3.2. *Let k be such that $c \cdot \log n \leqslant k \leqslant n/3$, where c is the constant from Theorem 3.2.1, and let* Enc *be the code from Theorem 3.2.1 that maps n bits to 2^k bits. Let $\mathcal{H}'$ be a set of k-input-bit Boolean functions such that $|\mathcal{H}'| \leqslant 2^{2n/3}$. Then, with probability at least $1 - 1/2^{n/2}$ over a random $x \in \{0,1\}^n$, the following holds for every $f \in \mathcal{H}'$:*

$$\mathop{\mathbb{P}}_{z \in \{0,1\}^k} [f(z) = \mathrm{Enc}(x)_z] \leqslant \frac{1}{2} + \frac{n}{2^{k/4}}. \tag{3.2}$$

Proof. The proof is by a counting argument. For every $f \in \mathcal{H}'$, consider the 2^k-bit string $\mathrm{tt}(f)$ which is the truth table computed by f. Let us say x is bad for f if Equation (3.2) does not hold, which means that $\mathrm{tt}(f)$ and $\mathrm{Enc}(x)$ agree on more than $1/2 + n/2^{k/4}$ positions. By the list-decodability of Enc, the number of such x's is at most $O\big(2^{k/2}/n\big)$. By an union bound over all the $2^{2n/3}$ functions in $\mathcal{H}'$, the fraction of bad x's is at most

$$\frac{O\big(2^{k/2}/n\big) \cdot 2^{2n/3}}{2^n} < \frac{1}{2^{n/2}},$$

as desired. □

We are now ready to prove Theorem 3.3.1.

Proof of Theorem 3.3.1. Let $A := A_{S,k}$ be the hard function on $2n$ variables defined as above, where S is the partition in the statement of the theorem, and let

$$B_j := \bigcup_{i \in [n/k]} \{z \colon z \text{ is the } j\text{-th variables of } S_i\}.$$

Also, let $\mathcal{H}'$ be the set of k-input-bit Boolean functions defined as follows:

$$\mathcal{H}' := \left\{ f \colon \exists h \in \mathcal{H} \text{ and } w \in \{0,1\}^k, \text{ such that } f(z) = h(z \oplus w) \text{ for all } z \in \{0,1\}^k \right\}.$$

That is, $\mathcal{H}'$ is the set of all possible "shifted" functions in $\mathcal{H}$. By Lemma 3.3.2, with probability at least $1 - 1/2^{n/2}$ over a random $x \in \{0,1\}^n$, for every $f \in \mathcal{H}'$ we have

$$\mathop{\mathbb{P}}_{z \in \{0,1\}^k} [f(z) = \mathrm{Enc}(x)_z] \leqslant \frac{1}{2} + \frac{n}{2^{k/4}}. \tag{3.3}$$

Let us call x *good* if it satisfies Equation (3.3).

To show the theorem, we need to upper bound the following probability, for

every circuit $C_0 \in \mathcal{C}_{2n}$ of size $s(n,k)$:

$$\begin{aligned}\mathop{\mathbb{P}}_{x,y\in\{0,1\}^n\times\{0,1\}^n}[A(x,y) = C_0(x,y)] &\leqslant \mathop{\mathbb{P}}_{x,y}[A(x,y) = C_0(x,y) \mid x \text{ is good}] \\ &\quad + \mathop{\mathbb{P}}_{x}[x \text{ is not good}] \\ &\leqslant \mathop{\mathbb{P}}_{x,y}[A(x,y) = C_0(x,y) \mid x \text{ is good}] + \frac{1}{2^{n/2}}.\end{aligned}$$

Let x be any fixed n-bit string that is good. Let $A'\colon \{0,1\}^n \to \{0,1\}$ be

$$A'(y) := A(x,y),$$

and let C be the circuit defined as

$$C(y) := C_0(x,y).$$

Note that since the class $\mathcal{C}$ is closed under restriction, C is a circuit from $\mathcal{C}_n$ with size at most $s(n,k)$. We will show that

$$\mathop{\mathbb{P}}_{y\in\{0,1\}^n}\left[A'(y) = C(y)\right] \leqslant \frac{1}{2} + \frac{n}{2^{k/4}}.$$

Let S_i be the block in the assumption of the theorem such that

$$\{C\restriction_\rho\}_{\rho\in\{0,1\}^{[n]\setminus S_i}} \subseteq \mathcal{H}.$$

We have

$$\mathop{\mathbb{P}}_{y\in\{0,1\}^n}\left[A'(y) = C(y)\right] = \mathop{\mathbb{P}}_{\rho\in\{0,1\}^{[n]\setminus S_i}, z\in\{0,1\}^k}\left[A'\restriction_\rho(z) = C\restriction_\rho(z)\right].$$

It suffices to upper bound

$$\mathop{\mathbb{P}}_{z\in\{0,1\}^k}\left[A'\restriction_\rho(z) = C\restriction_\rho(z)\right]$$

for every $\rho \in \{0,1\}^{[n]\setminus S_i}$. For the sake of contradiction, suppose for some ρ, we have

$$\frac{1}{2} + \frac{n}{2^{k/4}} < \mathop{\mathbb{P}}_{z\in\{0,1\}^k}\left[A'\restriction_\rho(z) = C\restriction_\rho(z)\right] = \mathop{\mathbb{P}}_{z\in\{0,1\}^k}\left[\mathrm{Enc}(x)_\alpha = C\restriction_\rho(z)\right], \tag{3.4}$$

where $\alpha \in \{0,1\}^k$ is

$$\alpha_j := \mathsf{Parity}\Big(\rho|_{B_j\setminus S_i}\Big) \oplus z_j,$$

and $\rho|_{B_j \setminus S_i}$ denotes the partial assignment given by ρ but restricted to only variables in the set $B_j \setminus S_i$. That is, α is some "shift" of z, so α is uniformly distributed for uniformly random z. Therefore, Equation (3.4) implies

$$\mathop{\mathbb{P}}_{z \in \{0,1\}^k} [\mathrm{Enc}(x)_z = C\restriction_\rho (z \oplus w)] > \frac{1}{2} + \frac{n}{2^{k/4}},$$

for some $w \in \{0,1\}^k$. This gives a function in $\mathcal{H}'$ that computes $\mathrm{Enc}(x)$ on more than $1/2 + n/2^{k/4}$ positions, which contradicts the assumption that x is good. □

3.4 #SAT Algorithms

In this section, we present our #SAT algorithm for comparator circuits. As mentioned briefly in Section 2.2, we will need a preprocessed data structure that enables us to efficiently convert a circuit with small number of wires but large number of gates to an equivalent circuit (with the same number of wires) whose number of gates is at most quadratic in the number wires.

3.4.1 Memorisation and simplification of comparator circuits

Lemma 3.4.1. *Let $n, \ell \geqslant 1$ be integers. For any fixed labelling of ℓ wires on n variables, there is a data structure* DS *such that*

- DS *can be constructed in time $2^n \cdot \ell^{O(\ell^2)}$.*
- *Given access to* DS *and given any comparator circuit C with ℓ wires (whose labelling is consistent with the one used for* DS*) and s gates, we can output in time* $\mathsf{poly}(s, \ell)$ *the number of satisfying assignments of C. Moreover, we obtain a comparator circuit with ℓ wires and at most $\ell \cdot (\ell - 1)/2$ gates that is equivalent to C.*

Proof. We know that every comparator circuit with ℓ wires has an equivalent circuit with $\ell \cdot (\ell - 1)/2$ gates (Lemma 3.1.4). Therefore, we can try to memorise the number of satisfying assignments for each of these circuits (by brute-force). Then for a given circuit C with ℓ wires and s gates where $s \gg \mathsf{poly}(\ell)$, we need to simplify C to be a circuit with $\ell \cdot (\ell - 1)/2$ gates so that we can look up its number of satisfying assignments, which was already computed. However, it is not clear how we can *efficiently* simplify such a comparator circuit.

The idea here is to remove the useless gates one by one (from left to right). To do this, firstly, we need to be able to tell whether a gate is useless, and secondly

whenever we remove a useless gate, we need to "re-wire" the gates that come after that gate, which can depend on the types of the useless gate that we are removing, as described in Proposition 3.1.2.

More specifically, DS will be a "tree-like" structure of depth at most $\ell \cdot (\ell - 1)/2 + 2$ where the internal nodes are labelled as gates. Note that a path from the root to any internal node in the tree gives a sequence of gates, which specifies a comparator circuit up to a choice of the output wire. We will require the label of every internal node to be a useful gate in the circuit specified by the path from the root to the node. In other words, each internal node will branch on all possible useful gates that could occur next in the circuit. Moreover, each leaf is either labelled as a useless gate, with respect to the circuit specified by the path from the root to the current leaf, or is labelled as a single wire that is designed to be the output wire.

For every leaf that is a useless gate, we store its type, and for each leaf that is a single wire, we store the number of satisfying assignments of the circuit that is specified by the path from the root to the leaf. Moreover, each internal node is a useful gate whose children are indexed by the set of all possible gates (each is an ordered pair of wires) and the set of wires (called an *output leaf*). Note that checking whether a new gate is useless and computing its type require evaluating the current circuit on all possible inputs, which takes time $2^n \cdot \mathsf{poly}(\ell)$, but this is fine with our running time. Similarly, we can compute the number of satisfying assignments in each output leaf by brute force. Note that by Theorem 3.1.3, the depth of such a tree is at most $\ell \cdot (\ell - 1)/2 + 2$, otherwise there would be a comparator circuit with ℓ wires that has more than $\ell \cdot (\ell - 1)/2$ useful gates. Since each internal node has at most ℓ^2 children, the tree has at most $\ell^{O\left(\ell^2\right)}$ nodes in total. Since each node can be constructed in time $2^n \cdot \mathsf{poly}(\ell)$, the running time is clear.

To look up the number of the satisfying assignments of a given circuit C (with a labelling of the wires that is consistent with the one used for DS), we start from the root of DS, and move down the tree as we look at the gates in C one by one (from left to right in the natural way). If we reach an output leaf, we output the number of satisfying assignments stored in that leaf. However, if we reach a leaf v that is specified as a useless gate, we remove the corresponding gate in C and update the gates that come after it according to the type of this useless gate, using Proposition 3.1.2. Once we update the circuit, we start again from the parent of v and look at the next gate in the updated circuit. We repeat this until we reach an output leaf. □

3.4.2 The algorithm

We will show an algorithm for comparator circuits with small number of wires, while the number of gates can be polynomial.

Theorem 3.4.2. *There is a constant $d > 1$ and a deterministic algorithm such that for every $k \leqslant n/4$, given a comparator circuit on n variables with at most*

$$\frac{n^{1.5}}{d \cdot k \cdot \sqrt{\log n}}$$

wires and $\mathsf{poly}(n)$ *gates, the algorithm outputs the number of satisfying assignments of C in time*

$$2^{n-k} \cdot \mathsf{poly}(n).$$

Proof. Let C be a comparator circuit with $\ell \leqslant n^{1.5}/\left(d \cdot k \cdot \sqrt{\log n}\right)$ wires and $\mathsf{poly}(n)$ gates, where $d \geqslant 1$ is a sufficiently large constant.

We partition the n variables almost-evenly into $\lfloor n/k \rfloor$ consecutive blocks, denoted as $S_1, S_2, \ldots, S_{\lfloor n/k \rfloor}$. We then count the number of wires that are labelled by the variables in each S_i and let

$$w_i := \left|\{u \colon u \text{ is a wire labelled by some } x \in S_i \text{ (or its negation)}\}\right|.$$

We have

$$\sum_{i \in [\lfloor n/k \rfloor]} w_i = \ell,$$

which implies that there is a particular $i \in [k]$ such that

$$w_i \leqslant \frac{\ell}{\lfloor n/k \rfloor} \leqslant \frac{1}{d} \cdot \sqrt{\frac{n}{\log n}} =: \ell_0.$$

Moreover, we can find such i efficiently.

Constructing DS**.** Using Lemma 3.4.1, we create a data structure DS with w_i wires and $|S_i| \leqslant k$ variables and a labelling consistent with that of C for the wires labelled by variables from S_i. This can be done in time

$$2^k \cdot \ell_0^{O(\ell_0^2)} = 2^{k + O\left(\ell_0^2 \cdot \log \ell_0\right)} \leqslant 2^{k + n/2}.$$

Enumeration. For each $\rho \in \{0,1\}^{[n] \setminus S_i}$, we obtain a restricted circuit $C \restriction_\rho$ (on either k or $k+1$ variables), which has ℓ_0 wires (whose labelling is consistent with the one used for DS created above) and has $\mathsf{poly}(n)$ gates. Then using DS, we can

efficiently look up the number of satisfying assignments of $C\restriction_\rho$. Finally we sum over these numbers over all such ρ's and this gives the number of satisfying assignments of C.

The total running time of the above algorithm is

$$2^{k+n/2} + 2^{n-k} \cdot \mathsf{poly}(n) = 2^{n-k} \cdot \mathsf{poly}(n),$$

as desired. □

3.5 Pseudorandom Generators and MCSP Lower Bounds

In this section, we show a PRG for small comparator circuits, and derive from it lower bounds for comparator circuits computing MCSP.

3.5.1 Proof of the PRG

We start with some definitions and notations.

- For a Boolean function f, we denote by $\ell(f)$ the minimum number of wires in a comparator circuit computing f.
- We will often describe a restriction $\rho \in \{0,1,*\}^n$ as a pair $(\sigma, \beta) \in \{0,1\}^n \times \{0,1\}^n$. The string σ is the characteristic vector of the set of coordinates that are assigned $*$ by ρ, and β is an assignment of values to the remaining coordinates. The string σ is also called a *selection*.
- We say that a distribution $\mathcal{D}$ on $\{0,1\}^n$ is a *p-regular* random *selection* if $\mathbb{P}_{\sigma\sim\mathcal{D}}[\sigma(i) = 1] = p$ for every $i \in [n]$.

As mentioned in Section 2.2, we will need a result saying that the number of wires in a comparator circuit shrinks with high probability under pseudorandom restrictions.

Lemma 3.5.1. *Let c be a constant and let $f : \{0,1\}^n \to \{0,1\}$. Let $\ell := \ell(f)$ and $p = \ell^{-2/3}$, and suppose that $\ell = n^{\Omega(1)}$. There exists a p-regular pseudorandom selection $\mathcal{D}$ over n variables that is samplable using $r = \mathsf{poly}\log(\ell)$ random bits such that*

$$\mathop{\mathbb{P}}_{\sigma\sim\mathcal{D},\,\beta\sim\{0,1\}^n}\left[\ell(f\restriction_{(\sigma,\beta)}) \geqslant 2^{3\sqrt{c\log\ell}} \cdot p\ell\right] \leqslant 2 \cdot \ell^{-c}.$$

Moreover, there exists a circuit of size $\mathsf{poly}\log(\ell)$ such that, given $j \in \{0,1\}^{\log n}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th coordinate of $\mathcal{D}(z)$.

The proof of Lemma 3.5.1 follows closely that of [IMZ19, Lemma 5.3], except for that here we also need to show that the pseudorandom restriction can be computed with small size circuits. Such a restriction is proved to exist in Lemma 18 of [CKLM20]. For completeness, a proof is presented in Section 3.5.3.

Theorem 3.5.2 (Local PRGs)**.** *For every $n \in \mathbb{N}$, $\ell = n^{\Omega(1)}$, and $\varepsilon \geqslant 1/\mathsf{poly}(n)$, there is a pseudorandom generator $G\colon \{0,1\}^r \to \{0,1\}^n$, with seed length*

$$r = \ell^{2/3+o(1)}$$

that ε-fools comparator circuits with ℓ wires over n variables. Moreover, for every seed $z \in \{0,1\}^r$, there is a circuit D_z of size $\ell^{2/3+o(1)}$ such that, given as input $j \in [n]$, D_z computes the j-th bit of $G(z)$.

Proof Sketch. In [IMZ19], it is shown that if a circuit class "shrinks" with high probability under a pseudorandom restriction, then we can construct pseudorandom generators for this circuit class with non-trivial seed-length. The authors of [CKLM20] then showed that if the same shrinkage property holds for random selections that can be efficiently sampled and computed, then we can obtain local PRGs. In Lemma 3.5.1, we proved exactly what is required by [CKLM20] to obtain local PRGs for comparator circuits.

More specifically, by following the proof of [CKLM20, Lemma 16], we can derive the theorem, adjusting the parameters there in a natural way. In particular, we will use $p := \ell^{-2/3}$ so that after the pseudorandom restriction in Lemma 3.5.1, the restricted comparator circuit has at most $\ell_0 := 2^{O(\sqrt{\log \ell})} \cdot p\ell = 2^{O(\sqrt{\log \ell})} \cdot \ell^{1/3}$ wires (with high probability). Another observation needed in the proof is that, by Lemma 3.1.4, there can be at most $2^{\ell^{2/3+o(1)}}$ distinct functions for comparator circuits with this many wires. We omit the details here. □

3.5.2 Proof of the MCSP lower bound

We prove the following stronger result which implies Theorem 1.2.4.

Theorem 3.5.3. *For every large enough $n \in \mathbb{N}$, any $\varepsilon > 0$ and any $0 < \alpha \leqslant 1 - \varepsilon$, $\mathsf{MCSP}[n^\alpha]$ on inputs of length n cannot be computed by comparator circuits with $n^{1+\alpha/2-\varepsilon}$ wires.*

Proof. Let f denote the function $\mathsf{MCSP}[n^\alpha]$ on inputs of length n. For the sake of contradiction, suppose f can be computed by a comparator circuit C with $n^{1+\alpha/2-\varepsilon}$ wires, for some $\varepsilon > 0$.

Let $k := n^{\alpha+\varepsilon/2}$. Consider an (almost-even) partition of the n variables into $\lfloor n/k \rfloor$ *consecutive* blocks, denoted as $S_1, S_2, \ldots, S_{\lfloor n/k \rfloor}$. Again, by an averaging argument, there is some $i \in [\lfloor n/k \rfloor]$ such that after fixing the values of the variables outside S_i, the number of wires in the restricted circuit is at most

$$\ell := n^{1+\alpha/2-\varepsilon}/\lfloor n/k \rfloor = n^{1.5\alpha-\varepsilon/2}.$$

Let ρ be a restriction that fixes the values of the variables outside S_i to be 0 and leaves the variables in S_i unrestricted. Let G be the PRG from Theorem 3.5.2 that has seed length $r := \ell^{2/3+o(1)}$ and $(1/3)$-fools comparator circuits with at most ℓ wires.

On the one hand, since $|S_i| \geqslant k$, then by a counting argument, for a uniformly random $x \in \{0,1\}^n$, the circuit size of the truth table given by $\rho \circ x$ is at least $k/(10 \log k) > n^\alpha$, with probability at least $1/2$. In other words,

$$\mathbb{P}_{x \in \{0,1\}^n}[f\restriction_\rho (x) = 1] \leqslant 1/2.$$

On the other hand, by the second item of Theorem 3.5.2, for any seed $z \in \{0,1\}^r$, the output of the PRG $G(z)$, viewed as a truth table, represents a function that can be computed by a circuit of size $\ell^{2/3+o(1)}$. Then knowing $i \in [n]$ (which can be encoded using $\log(n)$ bits), the truth table given by $\rho \circ G(z)$ has circuit size at most

$$\mathsf{poly}\log(n) + \ell^{2/3+o(1)} \leqslant n^\alpha.$$

This implies

$$\mathbb{P}_{z \in \{0,1\}^r}[C\restriction_\rho (G(z)) = 1] = 1,$$

which contradicts the security of G. □

3.5.3 Pseudorandom Shrinkage for Comparator Circuits: Proof of Lemma 3.5.1

We now give a proof of Lemma 3.5.1, combining the arguments of [IMZ19] for branching programs and formulas.

Technical tools. We will need a Chernoff-Hoeffding bounds for distributions with bounded independence from [SSS95] (Lemmas 2.3 in [IMZ19]). Recall that a distribution $\mathcal{D}$ on $[m]^n$ is *k-wise independent* if, for any set $A \subseteq [n]$ of size $|A| \leqslant k$, the random variables $\{\sigma(i) : i \in A\}$ are mutually independent when $\sigma \sim \mathcal{D}$.

Lemma 3.5.4 ([SSS95])**.** *Let $a_1, \ldots, a_n \in \mathbb{R}_+$ and let $m = \max_i a_i$. Suppose that $X_1, \ldots, X_n \in \{0,1\}$ are k-wise independent random variables with $\mathbb{P}[X_i = 1] = p$. Let $X = \sum_i a_i X_i$ and $\mu = \mathbb{E}[X] = p\sum_i a_i$. We have $\mathbb{P}[X \geqslant 2k(m+\mu)] \leqslant 2^{-k}$.*

Lemma 3.5.5 ([IMZ19, Lemma 2.4])**.** *Let $X_1, \ldots, X_n \in \{0,1\}$ be k-wise independent random variables with $\mathbb{P}[X_i = 1] = p$. Let $X = \sum_i X_i$ and $\mu = \mathbb{E}[X] = np$. We have $\mathbb{P}[X \geqslant k] \leqslant \mu^k/k!$.*

Shrinkage of comparator circuits under pseudorandom restrictions. We first show the following result for comparator circuits which is analogous to [IMZ19, Lemma 5.2] for branching programs.

Lemma 3.5.6. *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and let $H \subseteq [n]$. For $h \in \{0,1\}^H$, let ρ_h denote the restriction that sets the variables in H to h, and leaves the other variables free. We have $\ell(f) \leqslant 2^{|H|} \cdot \left(\max_{h\in\{0,1\}^H} \ell(f\restriction_{\rho_h}) + |H|\right)$.*

Proof. For $h \in \{0,1\}^H$, let $\mathbb{1}_h : x \mapsto \mathbb{1}\{x = h\}$. Clearly, $\mathbb{1}_h$ can be computed by a comparator circuit with $|H|$ wires. Since $f = \bigvee_{h\in\{0,1\}^H}(\mathbb{1}_h \wedge f\restriction_{\rho_h})$, the result follows. □

Lemma 3.5.7 (Reminder of Lemma 3.5.1)**.** *Let c be a constant and let $f : \{0,1\}^n \to \{0,1\}$. Let $\ell := \ell(f)$ and $p = \ell^{-2/3}$, and suppose that $\ell = n^{\Omega(1)}$. There exists a p-regular pseudorandom selection $\mathcal{D}$ over n variables that is samplable using $r = \mathsf{poly}\log(\ell)$ random bits such that*

$$\mathbb{P}_{\sigma\sim\mathcal{D},\,\beta\sim\{0,1\}^n}\left[\ell(f\restriction_{(\sigma,\beta)}) \geqslant 2^{3\sqrt{c\log\ell}} \cdot p\ell\right] \leqslant 2\cdot\ell^{-c}.$$

Moreover, there exists a circuit of size $\mathsf{poly}\log(\ell)$ such that, given $j \in \{0,1\}^{\log n}$ and a seed $z \in \{0,1\}^r$, the circuit computes the j-th coordinate of $\mathcal{D}(z)$.

Proof. First, we note that a k-wise independent random selection that can be efficiently sampled and computed with the required parameters is proved to exist in Lemma 18 of [CKLM20]. Henceforth, we let ρ be the random restriction described by the pair (σ, β).

Let C be a comparator circuit with ℓ wires computing f. Let $k = c \cdot \log\ell$. For $i \in [n]$, let w_i be the number of wires in C labelled with the variable x_i. Let $\alpha = \sqrt{c/\log\ell}$. We say that $i \in [n]$ is *heavy* if $w_i \geqslant p^{1-\alpha}\cdot\ell$ and *light* otherwise. Let $H \subseteq [n]$ be the set of heavy variables. We have $|H| \leqslant (1/p)^{1-\alpha}$. Let also $H(\rho) := H \cap \rho^{-1}(*)$. Let ρ' be a restriction such that $\rho'(x) = \rho(x)$ for $x \notin H(\rho)$

and which sets the variables in $H(\rho)$ so as to maximize $\ell(f\restriction_{\rho'})$. By Lemma 3.5.6, we have $\ell(f\restriction_{\rho}) \leqslant 2^{|H(\rho)|+1} \cdot \ell(f\restriction_{\rho'})$.

We now let $h = \lceil 3/2 \cdot c/\alpha \rceil$, and observe that

$$\mathop{\mathbb{P}}_{\rho}\left[\ell(f\restriction_{\rho}) \geqslant 2^{h+3}kp^{1-\alpha}s\right] \leqslant \mathop{\mathbb{P}}_{\rho}\left[|H(\rho)| \geqslant h\right] + \mathop{\mathbb{P}}_{\rho}\left[\ell(f\restriction_{\rho'}) \geqslant 4kp^{1-\alpha}\ell\right].$$

Let X_i be a random variable such that $X_i = 1$ iff $\rho(i) = *$. From Lemma 3.5.5, it follows that the first term can be bounded by $(|H|p)^h \leqslant p^{\alpha h} \leqslant \ell^{-c}$. For the second term, we can apply Lemma 3.5.4 on the light variables with $\mu \leqslant p\ell$ and $m < p^{1-\alpha}\ell$, so that $m + \mu \leqslant 2p^{1-\alpha}\ell$, thus bounding the probability by $2^{-k} \leqslant \ell^{-c}$. □

3.6 Learning Algorithms

Recall that a (distribution-independent) PAC learning algorithm for a class of functions $\mathcal{C}$ has access to labelled examples $(x, f(x))$ from an unknown function $f \in \mathcal{C}$, where x is sampled according to some (also unknown) distribution $\mathcal{D}$. The goal of the learner is to output, with high probability over its internal randomness and over the choice of random examples, a hypothesis h that is close to f under $\mathcal{D}$. As in [ST17], here we consider the stronger model of "randomised exact learning from membership and equivalence queries". It is known that learnability in this model implies learnability in the distribution-independent PAC model with membership queries (see [ST17, Section 2] and the references therein).

Theorem 3.6.1 ([ST17, Lemma 4.4])**.** *Fix any partition $S_1, S_2, \ldots, S_{n^{1-n^\delta}}$ of $[n]$ into equal-size subsets, where each S_i is of size n^δ and $\delta > 0$. Let $\mathcal{C}$ be a class of n-variate functions such that for each $f \in \mathcal{C}$, there is an S_i such that $\left|\{f\restriction_{\rho}\}_{\rho\in\{0,1\}^{[n]\setminus S_i}}\right| \leqslant 2^{n^\beta}$, where $\beta < 1$ and moreover $\delta + \beta < 1$. Then there is a randomised exact learning algorithms for $\mathcal{C}$ that uses membership and equivalence queries and runs in time $2^{n-n^\delta} \cdot \mathsf{poly}(n)$.*

Corollary 3.6.2. *For every $\varepsilon > 0$, there is a randomised exact learning algorithms for comparator circuits with $n^{1.5-\varepsilon}$ wires that uses membership and equivalence queries that runs in time $2^{n-n^{\Omega(\varepsilon)}} \cdot \mathsf{poly}(n)$.*

Proof. Consider Theorem 3.6.1 and any partition $S_1, S_2, \ldots, S_{n^{1-n^\delta}}$ of the n variables into equal-size subsets, each is of size n^δ, where $\delta := \varepsilon/3$. Then by an averaging argument, for every comparator circuit C with $n^{1.5-\varepsilon}$ wires, there is some S_i such that after fixing the variables outside of S_i, the number of wires in the restricted circuit is at most $\ell := n^{1.5-\varepsilon}/n^{1-\delta} \leqslant n^{.5-2\varepsilon/3}$. By Lemma 3.1.4, such a restricted

circuit computes some function that is equivalent to a circuit with $\ell(\ell-1)/2$ gates, and there are at most $\ell^{O(\ell^2)} \leqslant 2^{n^{1-\varepsilon/2}}$ such circuits. Therefore we have

$$\left|\{C\restriction_\rho\}_{\rho\in\{0,1\}^{[n]\setminus S_i}}\right| \leqslant 2^{n^\beta},$$

where $\beta := 1-\varepsilon/2 < 1$ and $\delta+\beta < 1$. The algorithm then follows from Theorem 3.6.1. □

Chapter 4

Constant-depth circuits vs. monotone circuits

Abstract

In this chapter, we establish new separations between the power of monotone and general (non-monotone) Boolean circuits:

- For every $k \geqslant 1$, there is a monotone function in AC^0 (constant-depth poly-size circuits) that requires monotone circuits of depth $\Omega(\log^k n)$. This significantly extends a classical result of Okol'nishnikova [Oko82] and Ajtai and Gurevich [AG87]. In addition, our separation holds for a monotone graph property, which was unknown even in the context of AC^0 versus mAC^0.
- For every $k \geqslant 1$, there is a monotone function in $\mathsf{AC}^0[\oplus]$ (constant-depth poly-size circuits extended with parity gates) that requires monotone circuits of size $\exp(\Omega(\log^k n))$. This makes progress towards a question posed by Grigni and Sipser [GS92].

These results show that constant-depth circuits can be more efficient than monotone formulas and monotone circuits when computing monotone functions.

In the opposite direction, we observe that non-trivial simulations are possible in the absence of parity gates: every monotone function computed by an AC^0 circuit of size s and depth d can be computed by a monotone circuit of size $2^{n-n/O(\log s)^{d-1}}$. We show that the existence of significantly faster monotone simulations would lead to breakthrough circuit lower bounds. In particular, if every monotone function in AC^0 admits a polynomial size monotone circuit, then NC^2 is not contained in NC^1.

Finally, we revisit our separation result against monotone circuit size and investigate the limits of our approach, which is based on a monotone lower bound for constraint satisfaction problems (CSPs) established by Göös, Kamath, Robere and Sokolov [GKRS19] via lifting techniques. Adapting results of Schaefer [Sch78] and Allender, Bauland, Immerman, Schnoor and Vollmer [ABI+09], we obtain an unconditional classification of the monotone

circuit complexity of Boolean-valued CSPs via their polymorphisms. This result and the consequences we derive from it might be of independent interest.

Organisation of the chapter

We begin with a preliminary discussion of definitions and results in Section 4.1, passing on to our new separations between constant-depth circuits and monotone circuits in Section 4.2. We will then discuss complexity-theoretic consequences from monotone simulations in Section 4.3, and finally discuss CSPs in Section 4.4. Deferred proofs of our CSP section will be given at Section 4.5. Some appendices to this chapter are also given: the reader is referred to Appendix B.1 for a proof of a superpolynomial monotone circuit lower bound for 3-XOR-SAT without using lifting theorems, and a table of clones in Appendix B.2.

4.1 Preliminaries

We review definitions, notation and results that will be used throughout the chapter. We remark that some definitions were already given at our introductory discussion in Sections 1.3 and 2.2. We refer the reader also to Appendix B.2, where a table of Boolean clones and their definitions is given.

4.1.1 Notation

Boolean functions. We denote by Mono the set of all monotone Boolean functions. We define $\mathsf{poly} = \{n \mapsto n^C : C \in \mathbb{N}\}$. A Boolean function $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ is said to be a graph property if $f(G) = f(H)$ for any two isomorphic graphs G and H. Let $\mathcal{F} = \{f_n\}_{n\in\mathbb{N}}$ be a sequence of graph properties, where f_n is defined over undirected graphs on n vertices. We say that $\mathcal{F}$ is *preserved under homomorphisms* if, whenever there is a homomorphism from a graph G to a graph H, we have $\mathcal{F}(G) \leqslant \mathcal{F}(H)$. We denote by $\mathsf{HomPreserving}$ the set of all graph properties which are preserved under homomorphisms. Note that $\mathsf{HomPreserving} \subseteq \mathsf{Mono}$.

Boolean circuits. We denote by $\mathsf{AC}^0_d[s]$ the family of Boolean functions computed by size-s, depth-d Boolean circuits with unbounded fan-in $\{\wedge, \vee\}$-gates and input literals from $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$. We write $\mathsf{AC}^0[s]$ as a shorthand for $\bigcup_{d=1}^{\infty} \mathsf{AC}^0_d[s]$, and AC^0 as a shorthand of $\mathsf{AC}^0[n^{O(1)}] = \mathsf{AC}^0[\mathsf{poly}]$. We will also refer to $\mathsf{AC}^0_d[\mathsf{poly}]$ by AC^0_d. We write $\mathsf{DNF}[s]$ to denote the family of Boolean functions computed by size-s DNFs, where size is measured by number of terms. We write $\mathsf{CNF}[s]$ analogously. We

write $\mathsf{SIZE}[s]$ to denote the family of Boolean functions computed by size-s circuits. We write $\mathsf{DEPTH}[d]$ to denote the family of Boolean functions computed by fan-in 2 circuits of depth d. We denote by $\mathsf{AC}^0[\oplus]$ the family of Boolean functions computed by polynomial-size AC^0 circuits with unbounded fan-in $\oplus$-gates.

We denote by L the family of Boolean functions computed by logspace machines, and by NL the family of Boolean functions computed by polynomial-time nondeterministic logspace machines. Moreover, we denote by $\oplus\mathsf{L}$ the family of Boolean functions computed by polynomial-time nondeterministic logspace machines with a *parity* acceptance condition (i.e., an input is accepted if the number of accepting paths is odd).

Circuit complexity. Given a circuit class $\mathcal{C}$, we write $\mathsf{m}\mathcal{C}$ to denote the monotone version of $\mathcal{C}$. Given a function f, we write $\mathsf{mSIZE}(f)$ to denote the size of the smallest monotone circuit computing f and $\mathsf{mDEPTH}(f)$ to denote the smallest depth of a fan-in 2 monotone circuit computing f. Given two Boolean functions f, g, we write $f \leqslant_m^{\mathsf{mProj}} g$ if there exists a many-one reduction from f to g in which each bit of the reduction is a monotone projection[1] of the input.

Miscellanea. Let $\alpha \in \{0,1\}^n$. We define $|\alpha|_1 := \sum_{i=1}^n \alpha_i$. We call $|\alpha|_1$ the *Hamming weight* of α. We let $\mathrm{supp}(\alpha) = \{i \in [n] : \alpha_i = 1\}$. We let $\mathrm{THR}_{k,n} : \{0,1\}^n \to \{0,1\}$ be the Boolean function such that $\mathrm{THR}_{k,n}(x) = 1 \iff |x|_1 \geqslant k$.

4.1.2 Background results

The next lemma, which is proved via a standard "guess-and-verify" approach, shows that nondeterministic logspace computations can be simulated by circuits of size 2^{n^ε} and of depth $d = O_\varepsilon(1)$.

Lemma 4.1.1 (Folklore; see, e.g., [AHM+08, Lemma 8.1])**.** *For all $\varepsilon > 0$, we have $\mathsf{NL} \subseteq \mathsf{AC}^0[2^{n^\varepsilon}]$.*

4.2 Constant-Depth Circuits vs. Monotone Circuits

In this section, we prove Theorems 1.3.1 and 1.3.2. For the upper bounds, we require the logspace graph connectivity algorithm due to [Rei05] and the $\oplus\mathsf{L}$ algorithm for solving linear systems over $\mathbb{F}_2$ due to [BDHM92], as well as the depth-reduction techniques of [AKR+01, AHM+08]. On the lower bounds side, our proofs

[1] A monotone projection is a projection without negations.

rely on previous monotone circuit and depth lower bounds from [BGW99, GKRS19]. In order to obtain a monotone formula lower bound for a graph property, we prove a graph padding lemma in Section 4.2.2.

4.2.1 A monotone size lower bound for a function in $\mathsf{AC}^0[\oplus]$

In this section, we prove Theorem 1.3.2. We first recall the monotone circuit lower bound of [GKRS19] and a depth-reduction lemma implicit in [AKR+01] and [OSS19], whose full proof we give below for completeness. We remark that similar arguments can be employed to prove Lemma 4.1.1, essentially by replacing the $\oplus$ gates by $\vee$ gates.

As explained in Section 2.2, in its strongest form the separation result from [GKRS19] can be stated as follows.

Theorem 4.2.1 ([GKRS19])**.** *There exists $\varepsilon > 0$ such that $\oplus\mathsf{L} \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{o(n^{\varepsilon})}]$. Moreover, this separation is witnessed by* 3-XOR-SAT*.*

Lemma 4.2.2 (Folklore; see, e.g., [AKR+01, OSS19])**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a $\oplus\mathsf{L}$ machine. For every $\delta > 0$, there exists an $\mathsf{AC}^0[\oplus]$ circuit of size $2^{n^{\delta}}$ that computes f.*

Proof. Let M be a $\oplus\mathsf{L}$-machine computing f. Without loss of generality, we may assume that each configuration in the configuration graph G of M is *time-stamped* – in other words, each configuration carries the information of the number of computational steps it takes to arrive at it.[2] We may also assume that every accepting computation takes exactly the same amount of time, which means that every path from the starting configuration v_{start} to the accepting configuration v_{accept} has the same length in the configuration graph. These assumptions imply that the configuration graph is *layered* (because a configuration with time-stamp t can only point to configurations with time-stamp $t+1$) and acyclic. Note that, for a fixed machine, the configuration graph can be computed from the input string using a projection.

Let $m = n^{O(1)}$ be the time that an accepting computation takes. We now show how to count (modulo 2) the number of accepting paths from v_{start} to v_{accept} with a depth-d $\mathsf{AC}^0[\oplus]$ circuit. First, choose $m^{1/d} - 1$ configurations $v_1, \ldots, v_{m^{1/d}-1}$ (henceforth called "checkpoints") from $V(G)$, such that the configuration v_i is at the level $i \cdot m^{1-1/d}$ in the configuration graph (i.e., it takes $i \cdot m^{1-1/d}$ time steps to

[2] Formally, we can define a $\oplus\mathsf{L}$-machine M' such that the configurations of M' are (C, t), where C is a configuration of M, and $t = 0, 1, \ldots, m = n^{O(1)}$ is a number denoting the time in which the configuration was achieved. A configuration (C, t) can only reach a configuration $(C', t+1)$ in the configuration graph of M'.

arrive at v_i). For convenience, we let $v_0 = v_{\mathsf{start}}$ and $v_{m^{1/d}} = v_{\mathsf{accept}}$. We then count the number of paths from from v_{start} to v_{accept} that go through $v_1, \ldots, v_{m^{1/d}-1}$, and sum over all possible choices of the checkpoints. Since the graph is layered and each path from v_0 to $v_{m^{1/d}}$ has length exactly m, there is only one choice of checkpoints that witnesses a given path from v_0 to $v_{m^{1/d}}$, so no path is counted twice in this summation. Letting $\#\mathsf{paths}(s, t, \ell)$ denote the number of paths between configurations s and t with distance exactly ℓ, we obtain

$$\#\mathsf{paths}(v_0, v_{m^{1/d}}, m) = \sum_{v_1,\ldots,v_{m^{1/d}-1}} \prod_{i=0}^{m^{1/d}-1} \#\mathsf{paths}(v_i, v_{i+1}, m^{1-1/d}).$$

The above calculation can be done in modulo 2 with an unbounded fan-in XOR gate (replacing the summation) and an unbounded fan-in AND gate (replacing the product). Note that the formula above is recursive. Repeating the same computation for calculating (modulo 2) the expression $\#\mathsf{paths}(v_i, v_{i+1}, m^{1-1/d})$ for each i, we obtain a depth-$2d$ $\mathsf{AC}^0[\oplus]$ circuit for calculating the number of paths from v_{start} to v_{accept} (modulo 2). Clearly, the total size of the circuit is $2^{O(m^{1/d} \cdot \log m)}$, which is smaller than 2^{n^δ} for a large enough constant d. □

We now restate Theorem 1.3.2 and prove it by combining Theorem 4.2.1 and Lemma 4.2.2 with a padding trick.

Theorem 1.3.2 (Polynomial-size constant-depth vs. larger monotone size)**.** *For every $k \geqslant 1$, we have $\mathsf{AC}^0[\oplus] \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{(\log n)^k}]$.*

Proof. By Theorem 4.2.1, there exists $\varepsilon > 0$ and a monotone function $f \in \oplus\mathsf{L}$ such that any monotone circuit computing f has size $2^{\Omega(n^\varepsilon)}$.

Let $\delta = \varepsilon/k$ and let $m = 2^{n^\delta}$. Let $g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be the Boolean function defined as $g(x, y) = f(x)$. Note that g is a function on $N := m + n = 2^{\Theta(n^\delta)}$ bits. By Lemma 4.2.2, there exists an $\mathsf{AC}^0[\oplus]$ circuit computing f of size $2^{n^\delta} = N^{O(1)}$. The same circuit computes g. On the other hand, any monotone circuit computing g has size $2^{\Omega(n^\varepsilon)} = 2^{\Omega((\log N)^{\varepsilon/\delta})} = 2^{\Omega((\log N)^k)}$. □

Remark 4.2.3 (About a separation with an $\mathsf{AC}^0[q]$ upper bound)**.** *The argument above gives a monotone Boolean function in $\mathsf{AC}^0[\oplus]$ with superpolynomial monotone circuit complexity, but there is nothing particular about the field $\mathbb{F}_2$ compared to $\mathbb{F}_q$ for other primes q. We now sketch how to obtain a monotone Boolean function computable by AC^0 circuits augmented with unbounded fan-in MOD_q gates[3] with the same lower bound as that of Theorem 1.3.2.*

[3] We define $\mathsf{MOD}_q : \{0,1\}^n \to \{0,1\}$ as $\mathsf{MOD}_q(x) = 1 \iff \sum_{i \in [n]} x_i = 0 \pmod q$.

The paper [GKRS19] shows that there exists a monotone Boolean function f computable by monotone span programs over $\mathbb{F}_q$ whose monotone circuit complexity is $2^{n^{\Omega(1)}}$. One can again use the equivalence between span programs over $\mathbb{F}_q$ and $\mathsf{MOD}_q\mathsf{L}$ machines [KW93] to obtain a $\mathsf{MOD}_q\mathsf{L}$ machine computing f[4], and the same argument employed in Lemma 4.2.2 allows us to obtain an AC^0 circuit with MOD_q gates of size 2^{n^ε} for any desired constant $\varepsilon > 0$.

4.2.2 A monotone depth lower bound for a graph property in AC^0

In this section, we prove Theorem 1.3.1. We prove moreover that the function that separates $\mathsf{AC}^0 \cap \mathsf{Mono}$ and mNC^i can be taken to be a graph property. We state our result in its full generality below.

Theorem 4.2.4. *We have $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mDEPTH}[(\log n)^i]$, for every $i \geqslant 1$. In particular, we have $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mNC}^i$.*

First, we recall a result of [BGW99], which proves monotone lower bounds for the following function. Let $\mathsf{OddFactor}_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be the function that accepts a given graph if it contains an *odd factor* – in other words, a spanning subgraph in which the degree of every vertex is odd. Babai, Gál and Wigderson [BGW99] proved the following result:

Theorem 4.2.5 ([BGW99])**.** *Any monotone formula computing $\mathsf{OddFactor}_n$ has size $2^{\Omega(n)}$, and any monotone circuit computing $\mathsf{OddFactor}_n$ has size $n^{\Omega(\log n)}$.*

The proof in [BGW99] is actually for the case of *bipartite graphs*, but it easily extends to general graphs, since the bipartite case reduces to the general case by a monotone projection. The formula lower bound stated above is slightly stronger because it makes use of asymptotically optimal lower bounds on the randomized communication complexity of DISJ_n [KS92], which were not available to [BGW99]. We remark that, with a different language, a monotone circuit lower bound for $\mathsf{OddFactor}$ is also implicitly proved in Feder and Vardi [FV98, Theorem 30].

We now recall an upper bound for $\mathsf{OddFactor}$, implicitly proved in an unpublished note due to Johannsen [Joh03].

Theorem 4.2.6 ([Joh03])**.** *We have $\mathsf{OddFactor} \in \mathsf{L}$.*

[4]The complexity class $\mathsf{MOD}_q\mathsf{L}$ is studied in [BDHM92], and their equivalence to span programs is stated in [KW93]. A $\mathsf{MOD}_q\mathsf{L}$ machine is a nondeterministic machine just like a $\oplus\mathsf{L}$ machine, but instead of accepting if the number of accepting paths is odd, it accepts if the number of accepting paths is different from 0 (mod q).

Proof. We first recall the following observation about the $\mathsf{OddFactor}$ function, which appears in different forms in the literature (see [Urq87, Lemma 4.1] or [Juk12, Lemma 18.16]; see also [Joh03, Proposition 1] for a different proof.)

Claim. *A graph G has an odd factor if and only if every connected component of G has an even number of vertices.*

Proof. If a graph G has an odd factor, we can conclude that every connected component of G has an even number of vertices from the well-known observation that in every graph there is an even number of vertices of odd degree.

Now suppose that every connected component of G has an even number of vertices. We will iteratively construct an odd factor F of G. We begin with the empty graph. We take any two vertices u, v in the same connected component of G which currently have even degree in F, and consider any path $P = (x_1, \ldots, x_k)$ between u and v, where $x_1 = u$ and $x_k = v$. If the edge $x_i x_{i+1}$ is currently in F, we remove $x_i x_{i+1}$ from F; otherwise, we add $x_i x_{i+1}$ to F. It's easy to check that, in every iteration of this procedure, only the vertices u and v have the parity of their degree changed in F; the degree of every other vertex stays the same (modulo 2). Since every connected component has an even number of vertices, this means that, eventually, every vertex in F will have odd degree. □

Now it's easy to check in logspace if every connected component of G has an even number of vertices using Reingold's algorithm for undirected connectivity [Rei05]. It suffices to check if, for every vertex v of G, the number of vertices reachable from v is odd. □

Now, if we only desire to obtain a function in AC^0 not computed by monotone circuits of depth $(\log n)^i$, we can follow the same argument of Theorem 1.3.2, using Lemma 4.1.1 instead of Lemma 4.2.2. In order to obtain moreover a monotone *graph property* witnessing this separation, we will need the following lemma, which enables us to obtain a graph property after "padding" a graph property. We defer the proof of this lemma to the end of this section.

Lemma 4.2.7. *Let $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ be a monotone graph property on graphs of n vertices. The following holds.*

1. *If $f \in \mathsf{NC}^i$ for some $i > 1$, then there exists a monotone graph property g on graphs of $N = 2^{(\log n)^i}$ vertices such that $g \in \mathsf{NC}^1$ and $f \leqslant_m^{\mathsf{mProj}} g$.*

2. *If* $f \in \mathsf{NL}$*, then for all* $\varepsilon > 0$ *there exists a monotone graph property* g *on graphs of* $N = 2^{n^\varepsilon}$ *vertices such that* g *can be computed by* AC^0 *circuits of size* $N^{2+o(1)}$ *and* $f \leqslant_m^{\mathsf{mProj}} g$*.*

3. *If* $f \in \oplus\mathsf{L}$*, then for all* $\varepsilon > 0$ *there exists a monotone graph property* g *on graphs of* $N = 2^{n^\varepsilon}$ *vertices such that* g *can be computed by* $\mathsf{AC}^0[\oplus]$ *circuits of size* $N^{2+o(1)}$ *and* $f \leqslant_m^{\mathsf{mProj}} g$*.*

We are now ready to prove Theorem 4.2.4.

Proof of Theorem 4.2.4. Fix $n \in \mathbb{N}$ and take an $\varepsilon < 1/i$. Observing that $\mathsf{L} \subseteq \mathsf{NL}$, from Theorem 4.2.6 and item (2) of Lemma 4.2.7 we conclude that there exists a monotone graph property f on $N = 2^{n^\varepsilon}$ vertices such that $f \in \mathsf{AC}^0$ and $\mathsf{OddFactor}_n \leqslant_m^{\mathsf{mProj}} f$. By Theorem 4.2.5, any monotone circuit computing f has depth $\Omega(n) = \Omega((\log N)^{1/\varepsilon}) \gg (\log N)^i$. □

Raz and Wigderson [RW92] observed that there exists a monotone function $f \in \mathsf{NC}^1 \setminus \mathsf{mNC}$. Using Lemma 4.2.7, we observe moreover that it's possible to obtain this separation with a monotone graph property.

Proposition 4.2.8. *We have* $\mathsf{NC}^1 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mNC}$*.*

Proof. Observing that $\mathsf{L} \subseteq \mathsf{NC}^2$, we conclude from Theorem 4.2.6 and item (1) of Lemma 4.2.7 that there exists a monotone graph property f on $N = 2^{(\log n)^2}$ vertices such that $f \in \mathsf{NC}^1$ and $\mathsf{OddFactor}_n \leqslant_m^{\mathsf{mProj}} f$. By Theorem 4.2.5, any monotone circuit computing f has depth $\Omega(n) = \Omega(2^{\sqrt{\log N}})$, which implies $f \notin \mathsf{mNC}$. □

4.2.3 Efficient monotone padding for graph properties

We will now prove Lemma 4.2.7. We first recall some low-depth circuits for computing threshold functions, which we will use to design a circuit for efficiently computing the adjacency matrix of induced subgraphs.

Theorem 4.2.9 ([HWWY94])**.** *Let* $d > 0$ *be a constant. The function* $\mathrm{THR}_{(\log n)^d, n}$ *can be computed by an* AC^0 *circuit of size* $n^{o(1)}$ *and depth* $d + O(1)$*.*

Theorem 4.2.10 ([AKS83])**.** *For every* $k \in [n]$*, the function* $\mathrm{THR}_{k,n}$ *can be computed by a circuit of depth* $O(\log n)$ *and size* $n^{O(1)}$*.*

Lemma 4.2.11. *There exists a circuit* C_n^k *with* $\binom{n}{2} + n$ *inputs and* $\binom{k}{2}$ *outputs which, when given as input an adjacency matrix of a graph* G *on* n *vertices and a characteristic vector of a set* $S \subseteq [n]$ *such that* $|S| \leqslant k$*, outputs the adjacency*

matrix of the graph $G[S]$, padded with isolated vertices when $|S| < k$. The circuit has constant-depth and size $n^{2+o(1)}$ when $k = \mathsf{poly}\log(n)$, and size $n^{O(1)}$ and depth $O(\log n)$ otherwise.

Proof. Let $\{x_{ij}\}_{i,j\in[n]}$ encode the adjacency matrix of G. Let $\alpha \in \{0,1\}^n$ be the characteristic vector of S. Let $i, j \in [k]$. Note that $\{i, j\} \in E(G[S])$ if and only if there exists $a, b \in [n]$ such that

- α_a is the i-th non-zero entry of α,
- α_b is the j-th non-zero entry of α, and
- $x_{ab} = 1$ (i.e., a and b are connected in G).

We first consider the case $k = \mathsf{poly}\log(n)$. In this case, the first two conditions can be checked with circuits of size $n^{o(1)}$ using Theorem 4.2.9. Therefore, we can compute if i and j are adjacent using $n^{2+o(1)}$ gates and constant depth. As there are at most $(\log n)^{O(1)}$ such pairs, we can output $G[S]$ with at most $n^{2+o(1)}$ gates.

For any k, the first two conditions can be checked with an NC^1 circuit by Theorem 4.2.10. Since there are at most n^2 pairs i, j, the entire adjacency matrix can be computed with a $O(\log n)$-depth and polynomial-size circuit. □

We are ready to prove Lemma 4.2.7.

Proof of Lemma 4.2.7. We first prove (1). Fix $n \in \mathbb{N}$ and let $N = 2^{(\log n)^i}$. For a graph G on N vertices such that $|E(G)| \leqslant \binom{n}{2}$, let G_{clean} be the graph obtained from G by removing isolated vertices from G one-by-one, in lexicographic order, until one of the following two conditions are satisfied: (1) there are no more isolated vertices in G_{clean}, *or* (2) G_{clean} has exactly n vertices. Let $g : \{0,1\}^{\binom{N}{2}} \to \{0,1\}$ be the monotone graph property defined as follows:

$$g(G) := \left(|E(G)| > \binom{n}{2}\right) \vee (|V(G_{\mathsf{clean}})| > n) \vee (f(G_{\mathsf{clean}}) = 1).$$

Note that g accepts a graph G if and only if at least one of the following three conditions are satisfied:

1. G has at most $\binom{n}{2}$ edges, G_{clean} has exactly n vertices and $f(G_{\mathsf{clean}}) = 1$, or
2. G has more than $\binom{n}{2}$ edges, or
3. G_{clean} has more than n vertices.

We observe that the monotonicity of g follows from the monotonicity of f. We also claim that g is a graph property. Indeed, the graph G_{clean} is the same (up to isomorphism), irrespective of the order according to which the isolated vertices are removed from G. Moreover, the function f is also a graph property. Because of this, all the three conditions above are preserved under isomorphisms.

We first observe that f is a monotone projection of g. Indeed, given a graph G on n vertices, we can easily construct by a monotone projection a graph G' on N vertices and at most $\binom{n}{2}$ edges such that $f(G) = g(G')$. We just let G' have a planted copy of G, and all other vertices are isolated. Then $G'_{\mathsf{clean}} = G$ (up to isomorphism) and $g(G') = f(G_{\mathsf{clean}}) = f(G)$.

We now show how to compute g in NC^1. Let $\{x_{ij}\}_{i,j\in[N]}$ be the input bits of g, corresponding to the adjacency matrix of a graph G. The circuit computes as follows.

1. If $|E(G)| > \binom{n}{2}$, accept the graph G.
2. Compute the characteristic vector $\alpha \in \{0,1\}^N$ of the set of all non-isolated vertices of G. If $|\alpha|_1 > n$, accept the graph G.
3. Compute G_{clean} and output $f(G_{\mathsf{clean}})$.

Note that checking if $|E(G)| > \binom{n}{2}$ can be done in NC^1 by Theorem 4.2.10. Moreover, for all $i \in [N]$, we have $\alpha_i = \bigvee_{j\in[N]} x_{ij}$, and therefore α_i can be computed by a circuit of depth $O(\log N)$ and $O(N)$ gates. In total, the vector α can be computed with $O(N^2)$ gates and $O(\log N)$ depth. Finally, we can check if $|\alpha|_1 > n$ in NC^1 with a threshold circuit.

For the final step, we compute G_{clean}. If $|\alpha|_1 = n$, note that $G_{\mathsf{clean}} = G[\mathrm{supp}(\alpha)]$. When $|\alpha|_1 < n$, then G_{clean} is $G[\mathrm{supp}(\alpha)]$ padded with isolated vertices. We can therefore compute G_{clean} with the circuit C^n_N of Lemma 4.2.11. Moreover, since $f \in \mathsf{NC}^i$, we have that f can be computed by a circuit of size $n^{O(1)} = N^{o(1)}$ and depth $O((\log n)^i) = O(\log N)$. Therefore, computing $f(G_{\mathsf{clean}})$ can be done in NC^1. Overall, we get that $g \in \mathsf{NC}^1$.

In order to prove (2), it suffices to modify the proof above. The modification can be briefly described as follows. We let $N = 2^{n^\varepsilon}$. Every time Lemma 4.2.11 is applied, we use the AC^0 circuit instead of the NC^1 circuit, since $n = \mathsf{poly}\log(N)$. This ammounts to $N^{2+o(1)}$ many gates with unbounded fan-in. Moreover, since by assumption $f \in \mathsf{NL}$, applying Lemma 4.1.1 we obtain an AC^0 circuit for f of size $2^{n^{\varepsilon/2}} = N^{o(1)}$, so we can compute $f(G_{\mathsf{clean}})$ in constant depth with $N^{o(1)}$ gates.

Finally, for (3) it suffices to apply the same argument used for (2), replacing an application of Lemma 4.1.1 by an application of Lemma 4.2.2. □

4.3 Non-Trivial Monotone Simulations and Their Consequences

In contrast to Section 4.2, in this section we observe that a non-trivial simulation of AC^0 circuits by monotone circuits is possible. This follows from a refined version of the switching lemma proved by Rossman [Ros17a]. As a proof of concept, we use this simulation result to reprove a well-known AC^0 lower bound for Majority.

In the second part of this section, we show that if much faster simulations are possible, then even stronger non-monotone circuit lower bounds follow. We also show that this implication is true even if the simulation only holds for *graph properties*. Monotone simulations for graph properties are motivated by a result of Rossman [Ros08a], which shows that very strong monotone simulations are possible for *homomorphism-preserving graph properties*. The lower bounds from monotone simulations are proved with the simulation result and padding argument used in the previous section (Lemmas 4.1.1 and 4.2.7).

4.3.1 A non-trivial simulation for bounded-depth circuits

The earliest monotone simulation result was proved for DNFs by Quine [Qui53].

Theorem 4.3.1 (Quine [Qui53])**.** *For all $s : \mathbb{N} \to \mathbb{N}$, we have $\mathsf{DNF}[s] \cap \mathsf{Mono} \subseteq \mathsf{mDNF}[s]$.*

Proof. If a given DNF computes a monotone Boolean function, simply removing the negative literals continues to compute the same function. □

Let $\mathsf{DT}_{\mathsf{size}}(f)$ denote the size of a smallest decision-tree computing f. We will need a result obtained by Rossman [Ros17a].

Theorem 4.3.2 ([Ros17a])**.** *If $f : \{0,1\}^n \to \{0,1\}$ is computable by an AC^0 circuit of depth d and size s, then $\mathsf{DT}_{\mathsf{size}}(f) = 2^{(1-1/O(\log s)^{d-1})n}$.*

Theorem 4.3.3. *Let $s : \mathbb{N} \to \mathbb{N}$ and $d \geqslant 1$. We have $\mathsf{AC}^0_d[s] \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[t]$, where $t = n \cdot 2^{n(1-1/O(\log s)^{d-1})}$. Moreover, this upper bound is achieved by monotone DNFs of size t/n.*

Proof. Let f be a monotone function computable by an AC^0 circuit of depth d and size s. By Theorem 4.3.2, there exists a decision tree of size $2^{(1-1/O(\log s)^{d-1})n}$

computing f. Therefore, there exists a DNF of the same size computing f, which can be taken to be monotone by by Theorem 4.3.1. This can be converted into a monotone circuit of size $n \cdot 2^{(1-1/O(\log s)^{d-1})n}$. □

We observe that it is possible to immediately deduce an AC^0 lower bound for Majority using this simulation theorem. Even though near-optimal lower bounds for Majority have been known for a long time [Hås86] and the proof of the main technical tool (Theorem 4.3.2) behind our simulation result is similar to the one used by [Hås86], the argument below illustrates how a monotone simulation can lead to non-monotone circuit lower bounds.

Corollary 4.3.4. *Any depth-d AC^0 circuit computing* Majority *must have size at least $2^{\Omega((n/\log n)^{1/(d-1)})}$.*

Proof. Note that Majority has $\binom{n}{n/2} = \Omega(2^n/\sqrt{n})$ minterms. Therefore, any monotone DNF computing Majority has size at least $\Omega(2^n/\sqrt{n})$. By Theorem 4.3.3, it follows that the size s of a depth-d AC^0 computing Majority satisfies the following inequality:

$$2^{n(1-1/O(\log s)^{d-1})} = \Omega(2^{n-\frac{1}{2}\log n}).$$

From this equation we obtain $s = 2^{\Omega((n/\log n)^{1/(d-1)})}$. □

4.3.2 Non-monotone lower bounds from monotone simulations

We now show that if monotone circuits are able to efficiently simulate non-monotone circuits computing monotone Boolean functions, then striking complexity separations follow. We also show a result of this kind for simulations of graph properties. We first prove a lemma connecting the simulation of AC^0 circuits with the simulation of NL machines.

Lemma 4.3.5. *For all constants $\varepsilon > 0$ and $C \geqslant 1$, if $\mathsf{AC}^0 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{O((\log n)^C)}]$, then $\mathsf{NL} \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{o(n^\varepsilon)}]$.*

Proof. We prove the contrapositive. Suppose that there exists $\varepsilon > 0$ such that $\mathsf{NL} \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{o(n^\varepsilon)}]$. This means that there exists a monotone function f such that $f \in \mathsf{NL}$ and any monotone circuit computing f has size $2^{\Omega(n^\varepsilon)}$.

Let $\delta = \varepsilon/(2C)$ and let $m = 2^{n^\delta}$. Let $g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be the Boolean function defined as $g(x,y) = f(x)$. Note that g is a function on $N := m + n = 2^{\Theta(n^\delta)}$ bits. By Lemma 4.1.1, there exists an AC^0 circuit computing f of size $2^{n^\delta} = N^{O(1)}$. Moreover, any monotone circuit computing g has size $2^{\Omega(n^\varepsilon)} = 2^{\Omega((\log N)^{\varepsilon/\delta})} = 2^{\Omega((\log N)^{2C})}$. □

Next, we recall the strongest known monotone circuit and formula lower bounds for a monotone function in NP.

Theorem 4.3.6 ([PR17])**.** $\mathsf{NP} \cap \mathsf{Mono} \not\subseteq \mathsf{mDEPTH}[o(n)]$.

Theorem 4.3.7 ([CKR20])**.** $\mathsf{NP} \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{o(\sqrt{n}/\log n)}]$.

We are now ready to state and prove our first result regarding new complexity separations from monotone simulations. Recall that obtaining explicit lower bounds against depth-3 AC^0 circuits of size $2^{\omega(n^{1/2})}$ is a major challenge in circuit complexity theory, while the best lower bound on the size of depth-4 AC^0 circuits computing a function in NP is currently $2^{\Omega(n^{1/3})}$ [Hås86]. Moreover, no strict separation is known in the following sequence of inclusions of complexity classes: $\mathsf{ACC} \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1 \subseteq \mathsf{L} \subseteq \mathsf{NL} \subseteq \oplus\mathsf{L} \subseteq \mathsf{NC}^2$. We show that efficient monotone simulations would bring new results in both of these fronts. (We stress that all lower bound consequences appearing below refer to separations against non-uniform circuits.)[5]

Theorem 4.3.8. *Let $\mathcal{C}$ be a class of circuits. There exists $\varepsilon > 0$ such that the following holds:*

1. *If $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$, then $\mathsf{NP} \not\subseteq \mathsf{AC}^0_3[2^{o(n)}]$.*
2. *If $\mathsf{AC}^0_4 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{NP} \not\subseteq \mathsf{AC}^0_4[2^{o(\sqrt{n}/\log n)}]$.*
3. *If $\mathcal{C} \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[2^{O(n^{\varepsilon})}]$, then $\mathsf{NC}^2 \not\subseteq \mathcal{C}$.*
4. *If $\mathsf{AC}^0 \cap \mathsf{Mono} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$, then $\mathsf{NC}^2 \not\subseteq \mathsf{NC}^1$.*

Proof. We will prove each item separately.

Proof of (1). Let us assume that $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$. Let f be the function of Theorem 4.3.6. For a contradiction, suppose that $f \in \mathsf{AC}^0_3[2^{o(n)}]$. Let $\alpha : \mathbb{N} \to \mathbb{N}$ be such that $\alpha(n) \to_n \infty$ and f has a depth-3 AC^0 circuit of size $2^{n/\alpha}$. Let $m = 2^{n/(10\cdot\alpha)}$ and let $g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be the function $g(x,y) = f(x)$. Let $N = n + m = (1 + o(1))2^{n/(10\cdot\alpha)}$. Clearly, the function g has a depth-3 AC^0 circuit of size $2^{n/\alpha} = N^{O(1)}$. Since g is monotone, we conclude from the assumption that g is computed by a polynomial-size monotone formula. Now, since $f(x) = g(x, 1^m)$, we obtain a monotone formula of size $N^{O(1)} = 2^{o(n)}$ for computing f, which contradicts the lower bound of Theorem 4.3.6.

[5] In other words, all upper bounds are *uniform*, but the lower bounds hold even for *non-uniform* circuits. Note that this is stronger than lower bounds for uniform circuits.

Proof of (2). Similar to the proof of item (1), but using Theorem 4.3.7 instead.

Proof of (3). Suppose that $\mathsf{NC}^2 \subseteq \mathcal{C}$. By Theorem 4.2.1, there exists a monotone function $f \in \mathsf{NC}^2$ on n bits and a number $\varepsilon > 0$ such that $f \notin \mathsf{mSIZE}[2^{o(n^{\varepsilon})}]$. Therefore, for any $\delta > 0$ such that $\delta < \varepsilon$, we have $f \notin \mathsf{mSIZE}[2^{O(n^{\delta})}]$. Since, by assumption, we have $f \in \mathsf{NC}^2 \subseteq \mathcal{C}$, we obtain $\mathcal{C} \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{O(n^{\delta})}]$.

Proof of (4). If $\mathsf{NC}^2 \subseteq \mathsf{NC}^1$, then, by item (3), we get $\mathsf{NC}^1 \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[2^{o(n^{\varepsilon})}]$. From Lemma 4.3.5, we obtain $\mathsf{AC}^0 \cap \mathsf{Mono} \not\subseteq \mathsf{mSIZE}[\mathsf{poly}]$. □

As a motivation to the ensuing discussion, we recall a result of Rossman, who showed that any homomorphism-preserving graph property computed by AC^0 circuits is also computed by monotone AC^0 circuits [Ros08a].

Theorem 4.3.9 ([Ros08a])**.** $\mathsf{AC}^0 \cap \mathsf{HomPreserving} \subseteq \mathsf{mDNF}[\mathsf{poly}]$.

This inspires the question of whether general graph properties can also be efficiently simulated by monotone circuits. We show that, if true, such simulations would imply strong complexity separations. Let us first recall an exponential monotone circuit lower bound for monotone graph properties, and we will be ready to state and prove our main result.

Theorem 4.3.10 ([AB87])**.** *There exists* $\varepsilon > 0$ *such that* $\mathsf{NP} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mSIZE}[2^{o(n^{\varepsilon})}]$.

Theorem 4.3.11. *Let* $\mathcal{C}$ *be a class of circuits. The following holds:*

1. *If* $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$*, then* $\mathsf{L} \not\subseteq \mathcal{C}$*.*
2. *If* $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mDEPTH}[o(\sqrt{n})]$*, where* n *denotes the number of input bits, then* $\mathsf{L} \not\subseteq \mathcal{C}$*.*
3. *If* $\mathsf{AC}^0 \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$*, then* $\mathsf{NP} \not\subseteq \mathsf{NC}^1$*.*

Proof. We will prove each item separately.

Proof of (1). Suppose that $\mathsf{L} \subseteq \mathcal{C}$. By Theorem 4.2.5, the monotone graph property $\mathsf{OddFactor}$ satisfies $\mathsf{OddFactor} \notin \mathsf{mSIZE}[\mathsf{poly}]$. Moreover, we have the upper bound $\mathsf{OddFactor} \in \mathsf{L}$ by Theorem 4.2.6. Since, by assumption, we have $\mathsf{OddFactor} \in \mathsf{L} \subseteq \mathcal{C}$, we obtain $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mSIZE}[\mathsf{poly}]$.

Proof of (2). Suppose that $\mathsf{L} \subseteq \mathcal{C}$. By Theorems 4.2.5 and 4.2.6, there exists a monotone graph property $f \in \mathsf{L}$ such that $f \notin \mathsf{mDEPTH}[o(\sqrt{n})]$. Since, by assumption, we have $f \in \mathsf{L} \subseteq \mathcal{C}$, we obtain $\mathcal{C} \cap \mathsf{Mono} \cap \mathsf{GraphProperties} \not\subseteq \mathsf{mDEPTH}[o(\sqrt{n})]$.

Proof of (3). Suppose that $\mathsf{NP} \subseteq \mathsf{NC}^1$. By Theorem 4.3.10, there exists a monotone graph property $f \in \mathsf{NC}^1$ such that $\mathsf{mSIZE}(f) = 2^{\Omega(n^\varepsilon)}$ for some $\varepsilon > 0$. Let $\delta = \varepsilon/2$. By Lemma 4.2.7 (Item 2), there exists a monotone graph property g on $N = 2^{n^\delta}$ vertices computed by an AC^0 circuit of size $N^{2+o(1)}$ such that f is a monotone projection of g. Theorem 4.3.10 implies that any monotone circuit computing f has size $2^{\Omega(n^\varepsilon)} = 2^{\Omega((\log N)^2)} = N^{\omega(1)}$. □

4.4 Monotone Complexity of Constraint Satisfaction Problems

In this section, we study the monotone complexity of Boolean-valued CSPs. Our goal is to classify which types of Boolean CSPs are hard for monotone circuit size and monotone circuit depth, eventually proving Theorems 1.3.4 and 1.3.5.

We will first spend some time recalling standard definitions and concepts in the theory of CSPs (Section 4.4.1), as well as a few results about CSPs that were proved in previous works [Sch78, Jea98, BCRV03, BCRV04, ABI+09] (Section 4.4.2). We will then prove Theorem 1.3.5 in Section 4.4.3, and we will finally prove Theorem 1.3.4 in Section 4.4.5 after proving some auxiliary results in Section 4.4.4.

4.4.1 Definitions

For a good introduction to the concepts defined below, we refer the reader to [BCRV03, BCRV04]. We also refer the reader to Section 1.3.3 for the definition of the family of functions $\mathsf{CSP\text{-}SAT}_S$, as well as the terms *constraint application, S-formula* and *satisfiable formula*.

We denote by $p_i^n : \{0,1\}^n \to \{0,1\}$ the i-th *projection function* on n variables, whose operation is defined as $p_i^n(x) = x_i$. For a set of Boolean functions B, we denote by $[B]$ the *closure* of B, defined as follows: a Boolean function f is in $[B]$ if and only if $f \in B \cup \{\text{Identity}\}$ or if there exists $g \in B$ and $h_1, \ldots, h_k$ such that $f = g(h_1, \ldots, h_k)$, where each h_i is either a projection function or a function from $[B]$. We can equivalently define $[B]$ as the set of all Boolean functions that can be computed by circuits using the functions of B as gates. Note that $[B]$ necessarily

contains an infinite number of Boolean functions, since $p_1^n \in [B]$ for every $n \in \mathbb{N}$; moreover, the constant functions are not necessarily in $[B]$. We say that B is a *clone* if $B = [B]$. A few prominent examples of clones are the set of all Boolean functions (equal to $[\{\wedge, \neg\}]$), monotone functions (equal to $[\{\wedge, \vee, 0, 1\}]$), and linear functions (equal to $[\{\oplus, 1\}]$).

Remark 4.4.1. *The set of all clones forms a lattice, known as* Post's lattice*, under the operations* $[A] \sqcap [B] := [A] \cap [B]$ *and* $[A] \sqcup [B] := [A \cup B]$*. From the next section onwards, we will refer to the clones defined in [BCRV03] (such as* I_0*,* I_1*, etc.), assuming the reader is familiar with them. For the unfamiliar reader, we refer to Appendix B.2 and Table B.2 and Fig. 4.1, which contain all the definitions of the clones we will need, as well as the entire Post's lattice in graphical representation.*

To avoid confusion, we will always refer to clones with normal-Roman *font (e.g.,* $\mathrm{S}_1, \mathrm{I}_0$*, etc).*

Let S be a finite set of Boolean relations. We denote by $\mathsf{CNF}(S)$ the set of all S-formulas. We denote by $\mathsf{COQ}(S)$ the set of all relations which can be expressed with the following type of formula φ:

$$\varphi(x_1, \ldots, x_k) = \exists y_1, \ldots, y_\ell \, \psi(x_1, \ldots, x_k, y_1, \ldots, y_\ell),$$

where $\psi \in \mathsf{CNF}(S)$. The relations in $\mathsf{COQ}(S)$ will also be referred as *conjunctive queries* over S. We denote by $\langle S \rangle$ the set of relations defined as $\langle S \rangle := \mathsf{COQ}(S \cup \{=\})$. If $S = \langle S \rangle$, we say that S is a *co-clone*. We define

$$\mathsf{CSP} = \{\mathsf{CSP\text{-}SAT}_S : S \text{ is a finite set of relations}\}.$$

We say that $\mathsf{CSP\text{-}SAT}_S$ is *trivial* if $\mathsf{CSP\text{-}SAT}_S$ is a constant function.

Let R be a k-ary Boolean relation and let $f : \{0,1\}^\ell \to \{0,1\}$ be a Boolean function. For $x \in R$ and $i \in [k]$, we denote by $x[i]$ the i-th bit of x.

Definition 4.4.2. *We say that* f *is a* polymorphism *of* R*, and* R *is* an invariant *of* f*, if, for all* $x_1, \ldots, x_\ell \in R$*, we have*

$$(f(x_1[1], \ldots, x_\ell[1]), f(x_2[2], \ldots, x_\ell[2]), \ldots, f(x_k[k], \ldots, x_\ell[k])) \in R.$$

We denote the set of all polymorphisms of R *by* $\mathsf{Pol}(R)$*. For a set of relations* S*, we denote by* $\mathsf{Pol}(S)$ *the set of Boolean functions which are polymorphisms of all the relations of* S*. For a set of Boolean functions, we denote by* $\mathsf{Inv}(B)$ *the set of all Boolean relations which are invariant under all functions of* B *(i.e.,* $\mathsf{Inv}(B) =$

$\{R : B \subseteq \mathsf{Pol}(R)\}$).

The following summarises the important facts about clones, co-clones and polymorphisms that are relevant to the study of CSPs [JCG97].

Lemma 4.4.3. *Let S and S' be sets of Boolean relations and let B and B' be sets of Boolean functions. We have*

(i) $\mathsf{Pol}(S)$ *is a clone and* $\mathsf{Inv}(B)$ *is a co-clone;*

(ii) If $S \subseteq S'$, *then* $\mathsf{Pol}(S') \subseteq \mathsf{Pol}(S)$;

(iii) If $B \subseteq B'$, *then* $\mathsf{Inv}(B') \subseteq \mathsf{Inv}(B)$;

(iv) $\mathsf{COQ}(\mathsf{COQ}(S)) = \mathsf{COQ}(S)$;

(v) If $S \subseteq S'$, *then* $\mathsf{COQ}(S) \subseteq \mathsf{COQ}(S')$;

(vi) $\mathsf{Inv}(\mathsf{Pol}(S)) = \langle S \rangle$;

(vii) $\mathsf{Pol}(\mathsf{Inv}(B)) = [B]$.

We now define different types of reductions. We say that a reduction is a *monotone OR-reduction* if every bit of the reduction is either constant or can be computed by a monotone disjunction on the input variables. We write $f \leqslant_m^{\mathsf{mOR}} g$ if there exists a many-one monotone OR-reduction from f to g. We also write $f \leqslant_m^{\mathsf{AC}^0} g$ if there exists a many-one AC^0 reduction from f to g, and $f \leqslant_m^{\mathsf{mNL}} g$ if there exists a many-one mNL reduction from f to g[6]. Unless otherwise specified, every reduction we consider will generate an instance of polynomial size on the length of the input.

Finally, we denote by OR^k and NAND^k the k-ary OR and NAND relations, respectively.

4.4.2 Basic facts about CSP-SAT

We state here basic facts about the CSP-SAT function. These facts are proved in the original paper of Schaefer [Sch78], as well as in later papers [Jea98, BCRV03, BCRV04, ABI+09].

[6] A many-one AC^0 (resp. mNL) reduction is one in which each bit of the reduction is either constant or can be computed with a polynomial-size AC^0 circuit (resp. monotone nondeterministic branching program). Recall that a *monotone nondeterministic branching program* is a directed acyclic graph G with two distinguished vertices s and t, in which each edge e is labelled with an input function $\rho_e \in \{1, x_1, \ldots, x_n\}$. Given an input x, the program accepts if there exists a path from s to t in the subgraph G_x of G in which an edge e appears if $\rho_e(x) = 1$.

Lemma 4.4.4 below is one of the most important lemmas of this section and will be used many times. It states that $\mathsf{Pol}(S)$ characterises the monotone complexity of $\mathsf{CSP\text{-}SAT}_S$, in the sense that the sets of relations with few polymorphisms give rise to the hardest instances of CSPs. A non-monotone version of this result was proved in [Jea98, BCRV04, Theorem 2.4], and we check in Section 4.5.2 that their proofs also hold in the monotone case.

Lemma 4.4.4 (Polymorphisms characterise the complexity of CSPs [Jea98, BCRV04, Theorem 2.4])**.** *If* $\mathsf{Pol}(S_2) \subseteq \mathsf{Pol}(S_1)$*, then* $\mathsf{CSP\text{-}SAT}^n_{S_1} \leqslant^{\mathsf{mNL}}_m \mathsf{CSP\text{-}SAT}^{\mathsf{poly}(n)}_{S_2}$.

Theorem 4.4.5 gives monotone circuit upper bounds for some instances of $\mathsf{CSP\text{-}SAT}_S$. Non-monotone variants of this upper bound were originally obtained in the seminal paper of Schaefer [Sch78], and we again check that the monotone variants work in Section 4.5.3.

Theorem 4.4.5 (Monotone version of the upper bounds for CSP-SAT [Sch78, ABI+09])**.** *Let S be a finite set of relations. The following holds.*

1. *If* $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$ *or* $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$*, then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mSIZE}[\mathsf{poly}]$.
2. *If* $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$*, or* $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$*, or* $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$*, then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL}$.

We now recall the definitions of a few well-known CSPs. We define $\mathsf{3\text{-}Horn\text{-}SAT}_n : \{0,1\}^{2n^3+n} \to \{0,1\}$ as

$$\mathsf{3\text{-}Horn\text{-}SAT}_n = \mathsf{CSP\text{-}SAT}^n_{\mathcal{H}^3}, \text{ where}$$
$$\mathcal{H}^3 = \{(\neg x_1 \vee \neg x_2 \vee x_3), (\neg x_1 \vee \neg x_2 \vee \neg x_3), (x)\}\,.$$

Let also $\mathsf{3\text{-}AntiHorn\text{-}SAT} = \mathsf{CSP\text{-}SAT}_{\mathcal{A}^3}$, where

$$\mathcal{A}^3 = \{(x_1 \vee x_2 \vee \neg x_3), (x_1 \vee x_2 \vee x_3), (\neg x)\}\,.$$

Finally, let $\mathsf{2\text{-}SAT} = \mathsf{CSP\text{-}SAT}_\Gamma$, where $\Gamma = \{(x_1 \vee x_2), (x_1 \vee \neg x_2), (\neg x_1 \vee \neg x_2)\}$. We now observe that the set of polymorphisms of a given set of relations S contains E_2, V_2, or D_2 if, and only if, the relations in S can be described by a $\mathcal{H}^3$-formula (also known as *Horn* formula), $\mathcal{A}^3$-formula (also known as *anti-Horn* formula), or Γ-formula (also known as *bijunctive* formula) – respectively.

Lemma 4.4.6 ([CKS01, Lemmas 4.8 and 4.9])**.** *Let S be a finite set of relations. The following holds.*

1. $\mathrm{E}_2 \subseteq \mathsf{Pol}(S) \iff S \subseteq \mathsf{COQ}(\mathcal{H}^3)$;

2. $\mathrm{V}_2 \subseteq \mathsf{Pol}(S) \iff S \subseteq \mathsf{COQ}(\mathcal{A}^3)$;

3. $\mathrm{D}_2 \subseteq \mathsf{Pol}(S) \iff S \subseteq \mathsf{COQ}(\Gamma)$.

Finally, we state here a result of [ABI+09], which classifies the *non-monotone* complexity of $\mathsf{CSP\text{-}SAT}_S$ under $\leqslant_m^{\mathsf{AC}^0}$ reductions. The classification of the complexity of $\mathsf{CSP\text{-}SAT}_S$ is based solely on $\mathsf{Pol}(S)$. See Figure 4.1 for a graphical representation.

Theorem 4.4.7 (Refined classification of CSP problems [ABI+09, Theorem 3.1])**.** *Let S be a finite set of Boolean relations. The following holds.*

- *If* $\mathrm{I}_0 \subseteq \mathsf{Pol}(S)$ *or* $\mathrm{I}_1 \subseteq \mathsf{Pol}(S)$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is trivial.*
- *If* $\mathsf{Pol}(S) \in \{\mathrm{I}_2, \mathrm{N}_2\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leqslant_m^{\mathsf{AC}^0}$*-complete for* NP.
- *If* $\mathsf{Pol}(S) \in \{\mathrm{V}_2, \mathrm{E}_2\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leqslant_m^{\mathsf{AC}^0}$*-complete for* P.
- *If* $\mathsf{Pol}(S) \in \{\mathrm{L}_2, \mathrm{L}_3\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leqslant_m^{\mathsf{AC}^0}$*-complete for* $\oplus\mathsf{L}$.
- *If* $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S) \subseteq {\mathrm{S}_{00}}^2$ *or* $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S) \subseteq {\mathrm{S}_{10}}^2$ *or* $\mathsf{Pol}(S) \in \{\mathrm{D}_2, \mathrm{M}_2\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leqslant_m^{\mathsf{AC}^0}$*-complete for* NL.
- *If* $\mathsf{Pol}(S) \in \{\mathrm{D}_1, \mathrm{D}\}$*, then* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leqslant_m^{\mathsf{AC}^0}$*-complete for* L.
- *If* $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ *or* $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$*, then either* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0$ *or* $\mathsf{CSP\text{-}SAT}_S$ *is* $\leqslant_m^{\mathsf{AC}^0}$*-complete for* L.

4.4.3 A monotone dichotomy for CSP-SAT

In this section, we prove Theorem 1.3.5. We first prove Part (1) of the theorem (the dichotomy for circuit size), and then we prove Part (2) of the theorem (the dichotomy for circuit depth).

Dichotomy for circuits. To prove the dichotomy for circuits, we first show that, for any set of relations S whose set of polymorphisms is contained in L_3, we can monotonically reduce 3-XOR-SAT to $\mathsf{CSP\text{-}SAT}_S$.

Lemma 4.4.8. *Let S be a finite set of relations. If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$*, then we get* $\text{3-XOR-SAT} \leqslant_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}_S$.

Proof. Inspecting Post's lattice (Figure 4.1), note that the only clones strictly contained in L_3 are $\mathrm{L}_2, \mathrm{N}_2$ and I_2. We will first show that the reduction holds for the case $\mathsf{Pol}(S) = \mathrm{L}_2$ and then prove that the reduction also holds for the case $\mathsf{Pol}(S) = \mathrm{L}_3$. Lemma 4.4.4 will then imply the cases $\mathsf{Pol}(S) \in \{\mathrm{N}_2, \mathrm{I}_2\}$, since $\mathrm{I}_2 \subseteq \mathrm{N}_2 \subseteq \mathrm{L}_3$.

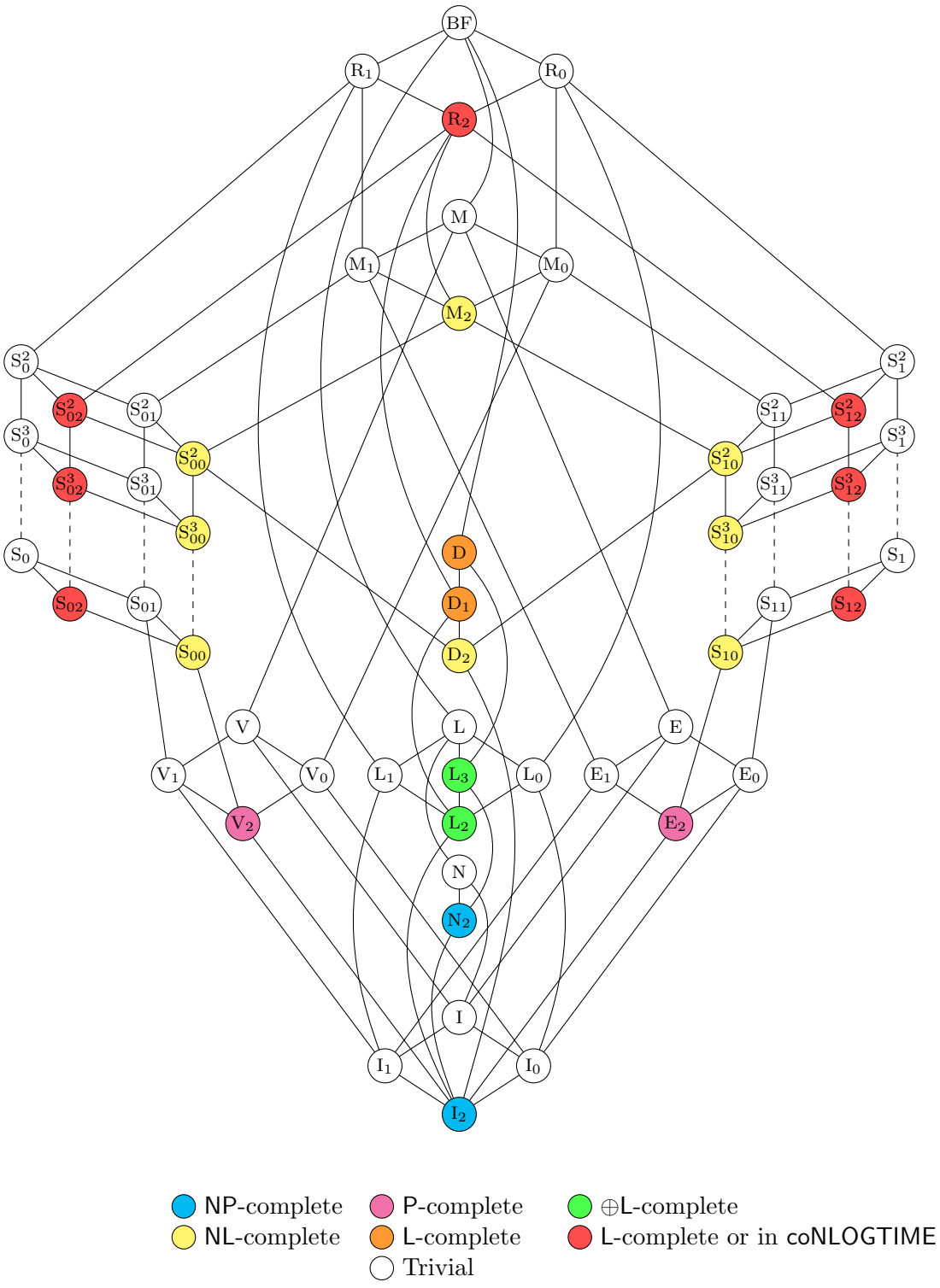


Figure 4.1: Graph of all closed classes of Boolean functions. The vertices are colored with the complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex. Trivial CSPs are those that correspond to constant functions. Every hardness result is proved under $\leqslant_m^{\mathsf{AC}^0}$ reductions. See Theorem 4.4.7 for details. A similar figure appears in [ABI+09, Figure 1].

It's not hard to check that, if $\mathsf{Pol}(S) = \mathrm{L}_2$, then $\mathsf{Pol}(S) \subseteq \mathsf{Pol}(\text{3-XOR-SAT})$ (it suffices to observe that bitwise XORing three satisfying assignments to a linear equation gives rise to a new satisfying assignment to the same equation). Therefore, from Lemma 4.4.4 we deduce that 3-XOR-SAT admits a reduction to $\mathsf{CSP\text{-}SAT}_S$ in mNL. In order to prove the case $\mathsf{Pol}(S) = \mathrm{L}_3$, we first prove the following claim.

Claim ([ABI+09, Lemma 3.11])**.** *Let S be a finite set of relations such that $\mathsf{Pol}(S) = \mathrm{L}_2$. There exists a finite set of relations S' such that $\mathsf{Pol}(S') = \mathrm{L}_3$ and, moreover, we have $\mathsf{CSP\text{-}SAT}^n_S \leqslant^{\mathsf{mProj}}_m \mathsf{CSP\text{-}SAT}^{n+1}_{S'}$.*

Proof. We describe the proof of Lemma 3.11 in [ABI+09] and observe that it gives a monotone reduction.

For a relation $R \in S$, let $R' = \{(\neg x_1, \ldots, \neg x_k) : (x_1, \ldots, x_k) \in R\}$. Let also $S' = \{R' : R \in S\}$. It's not hard to check that $\mathsf{Pol}(S') = \mathrm{L}_3$, since S' is an invariant of L_2 and N_2, and L_3 is the smallest clone containing both L_2 and N_2; moreover, if $\rho \in \mathsf{Pol}(S')$ and ρ is a Boolean function on at least two bits, then $\rho \in \mathsf{Pol}(S) = \mathrm{L}_2$.

Now let F be a instance of $\mathsf{CSP\text{-}SAT}^n_S$. For every constraint $C = R(x_1, \ldots, x_k)$ in F, we add the constraint $C' = R'(\alpha, x_1, \ldots, x_k)$ to the S'-formula F', where α is a new variable. Note that F' is a S'-formula, defined on $n+1$ variables, which is satisfiable if and only if F is satisfiable. Moreover, the construction of F' from F can be done with a monotone projection. □

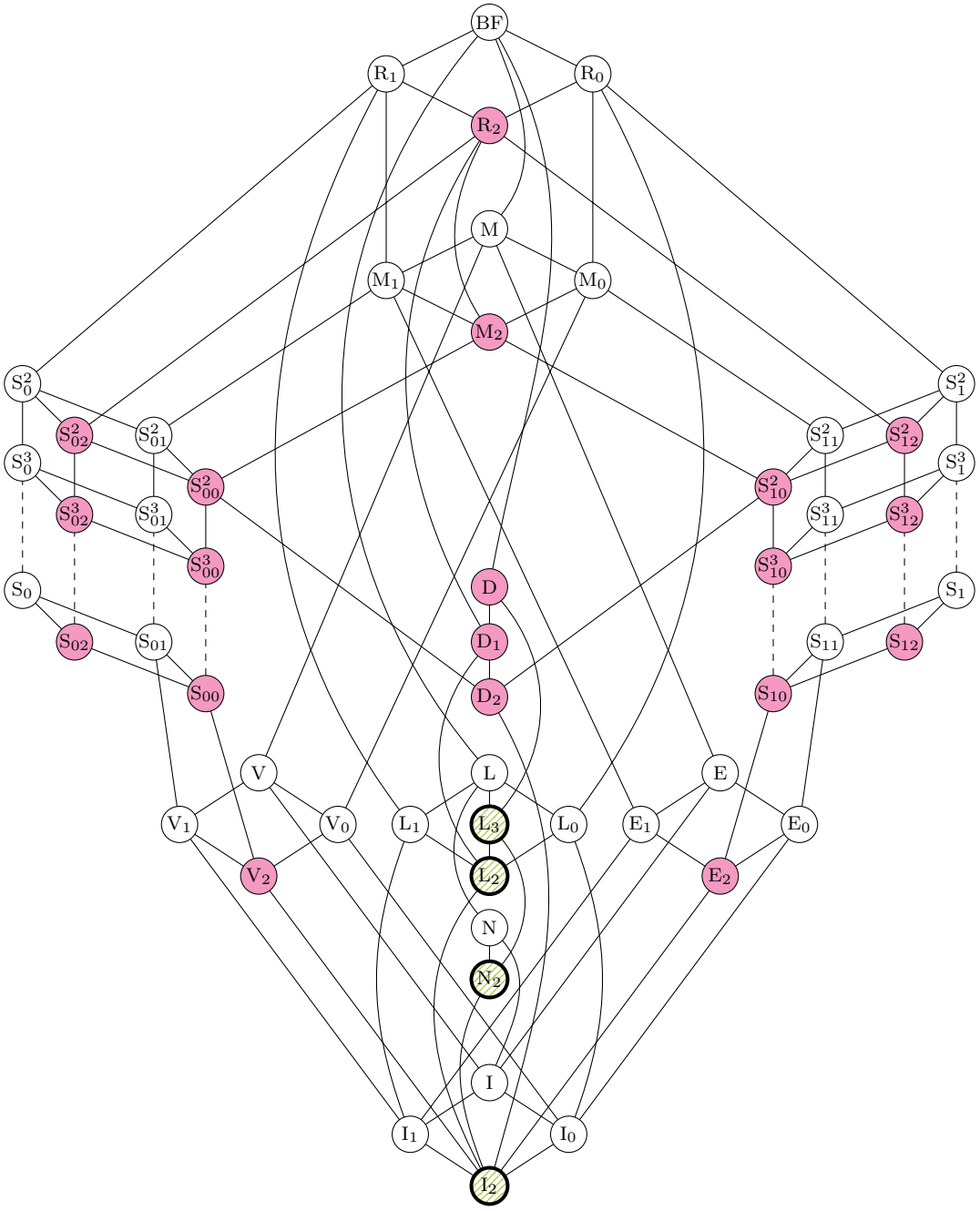
Since the case $\mathsf{Pol}(S) = \mathrm{L}_2$ holds, the case $\mathsf{Pol}(S) = \mathrm{L}_3$ now follows from Lemma 4.4.4 and the Claim. Finally, from Lemma 4.4.4 we conclude that the reduction also holds for the case $\mathsf{Pol}(S) \in \{\mathrm{N}_2, \mathrm{I}_2\}$, since $\mathrm{I}_2 \subseteq \mathrm{N}_2 \subseteq \mathrm{L}_3$. □

Theorem 4.4.9 (Dichotomy for monotone circuits)**.** *Let S be a finite set of relations. If $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$ then there is a constant $\varepsilon > 0$ such that $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = 2^{\Omega(n^\varepsilon)}$. Otherwise, we have $\mathsf{mSIZE}(\mathsf{CSP\text{-}SAT}_S) = n^{O(1)}$.*

Proof. If $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$, the lower bound follows from the 'moreover' part of Theorem 4.2.1, and Lemma 4.4.8. For the upper bound, we inspect Post's lattice (Figure 4.1). Observe that, if $\mathsf{Pol}(S) \not\subseteq \mathrm{L}_3$, the following are the only possible cases:

1. $\mathrm{I}_0 \subseteq \mathsf{Pol}(S)$ or $\mathrm{I}_1 \subseteq \mathsf{Pol}(S)$. In both cases, any $\mathsf{CNF}(S)$ is trivially satisfiable.
2. $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$ or $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$. In this case, $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mSIZE}[\mathsf{poly}]$ by Theorem 4.4.5.
3. $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$. In this case, $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL} \subseteq \mathsf{mSIZE}[\mathsf{poly}]$ by Theorem 4.4.5.

□



Solvable in mSIZE[poly].
Requires monotone circuits of size $2^{\Omega(n^{\varepsilon})}$.
Trivial (i.e., a constant function).

Figure 4.2: Illustration of Theorem 4.4.9. The vertices are colored with the *monotone circuit size* complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex.

Remark 4.4.10. *We remark that the lifting theorem of [GKRS19] (which is an ingredient in the proof of Theorem 4.2.1) is only used to prove that the monotone complexity of* $\mathsf{CSP\text{-}SAT}_S$ *is exponential when* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$*. If we only care to show a superpolynomial separation, then it suffices to apply the superpolynomial lower bound for CSPs with counting proved in [FV98, BGW99] using the approximation method. Indeed, we give an explicit proof in Appendix B.1. The same holds for the consequences of this theorem (see Theorem 4.4.20).*

Dichotomy for formulas. The following is proved in [RM99, GKRS19].

Theorem 4.4.11 ([RM99, GKRS19])**.** *There exists* $\varepsilon > 0$ *such that* $\mathsf{3\text{-}Horn\text{-}SAT} \in \mathsf{mSIZE}[\mathsf{poly}] \setminus \mathsf{mDEPTH}[o(n^{\varepsilon})]$.

Proof sketch. Since $\mathrm{E}_2 \subseteq \mathsf{Pol}(\mathcal{H}^3)$ (Lemma 4.4.6) and $\mathsf{3\text{-}Horn\text{-}SAT} = \mathsf{CSP\text{-}SAT}_{\mathcal{H}^3}$, the upper bound follows from Theorem 4.4.5. The lower bound follows from a lifting theorem of [RM99, GKRS19]. They show that the monotone circuit-depth of 3-Horn-SAT is at least the depth of the smallest Resolution-tree refuting a so-called *pebbling formula*. Since this formula requires Resolution-trees of depth n^{ε}, the lower bound follows. □

Analogously to the previous section, we show that 3-Horn-SAT reduces to $\mathsf{CSP\text{-}SAT}_S$ whenever $\mathsf{Pol}(S)$ is small enough, in a precise sense stated below. We then deduce the dichotomy for formulas with a similar argument.

Lemma 4.4.12. *Let* S *be a finite set of relations. If* $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$ *or* $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$*, then* $\mathsf{3\text{-}Horn\text{-}SAT} \leqslant_m^{\mathsf{mNL}} \mathsf{CSP\text{-}SAT}_S$.

Proof. We first consider the case $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$. Note that $\mathrm{E}_2 \subseteq \mathsf{Pol}(\mathsf{3\text{-}Horn\text{-}SAT})$ (Lemma 4.4.6). Therefore, from Lemma 4.4.4 we deduce that 3-Horn-SAT admits a reduction to $\mathsf{CSP\text{-}SAT}_S$ in mNL.

Observe that a $\mathcal{H}^3$-formula φ is satisfiable if and only if the $\mathcal{A}^3$-formula $\varphi(\neg x_1, \ldots, \neg x_n)$ is satisfiable. Therefore, we have $\mathsf{3\text{-}Horn\text{-}SAT} \leqslant_m^{\mathsf{mProj}} \mathsf{3\text{-}AntiHorn\text{-}SAT}$. Observing that $\mathrm{V}_2 \subseteq \mathsf{Pol}(\mathcal{A}^3)$ (see Lemma 4.4.6), the result now follows from Lemma 4.4.4 and the previous paragraph. □

Theorem 4.4.13 (Dichotomy for monotone formulas)**.** *Let* S *be a finite set of relations. If* $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$*, or* $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$*, or* $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$*, then there is a constant* $\varepsilon > 0$ *such that* $\mathsf{mDEPTH}(\mathsf{CSP\text{-}SAT}_S) = \Omega(n^{\varepsilon})$*. Otherwise, we have* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL} \subseteq \mathsf{mNC}^2 \subseteq \mathsf{mDEPTH}[\log^2 n]$.

Proof. We will first prove the lower bound. If $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$, the lower bound follows from Theorem 4.4.9. If $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$ or $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$, the lower bound follows from Theorem 4.4.11 and Lemma 4.4.12.

By inspecting Post's lattice (Remark 4.4.1), we see that the remaning cases are:

1. $\mathrm{I}_0 \subseteq \mathsf{Pol}(S)$ or $\mathrm{I}_1 \subseteq \mathsf{Pol}(S)$. In both cases, any $\mathsf{CNF}(S)$ is trivially satisfiable.
2. $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$, or $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$, or $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$. In all of those three cases, we have $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL}$ by Theorem 4.4.5. □

4.4.4 Some auxiliary results

In this section, we prove auxiliary results needed in the proof of a more general form of Theorem 1.3.4. In particular, we will prove that all $\mathsf{CSP\text{-}SAT}_S$ which are in AC^0 are also contained in $\mathsf{mAC}^0 \subseteq \mathsf{mNC}^1$. Moreover, we show that, if $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^1$, then $\mathsf{CSP\text{-}SAT}_S$ is L-hard under $\leqslant_m^{\mathsf{AC}^0}$ reductions.

We first observe that, when $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, there exists an efficient low-depth reduction from $\mathsf{CSP\text{-}SAT}_{S_1}$ to $\mathsf{CSP\text{-}SAT}_{S_2}$. This reduction, which will be useful in this section, is more refined than the one given by Lemma 4.4.4. A proof of the non-monotone version of this statement is found in [BCRV04, Proposition 2.3], and we give a monotone version of this proof in Section 4.5.2.

Lemma 4.4.14 ([BCRV04, Proposition 2.3])**.** *Let S_1 and S_2 be finite sets of relations. If $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, then there exists a constant $C \in \mathbb{N}$ such that $\mathsf{CSP\text{-}SAT}^n_{S_1} \leqslant_m^{\mathsf{mOR}} \mathsf{CSP\text{-}SAT}^{Cn}_{S_2}$.*

Proof. We defer the proof to Section 4.5.2. □

We now recall some lemmas from [ABI+09], and prove a few consequences from them. We say that a set S of relations *can express equality* if $\{=\} \subseteq \mathsf{COQ}(S)$.

Lemma 4.4.15 ([ABI+09])**.** *Let S be a finite set of relations. Suppose $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$ ($\mathrm{S}_{12} \subseteq \mathsf{Pol}(S)$, resp.) and that S cannot express equality. Then there exists $k \geqslant 2$ such that $S \subseteq \mathsf{COQ}(\{\mathsf{OR}^k, x, \neg x\})$ ($S \subseteq \mathsf{COQ}(\{\mathsf{NAND}^k, x, \neg x\})$, resp.).*

Proof. Follows from the proof of Lemma 3.8 of [ABI+09]. □

Lemma 4.4.16. *Let S be a finite set of relations such that $\mathsf{Pol}(S) \subseteq \mathrm{R}_2$. If $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S)$, and S cannot express equality, then $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mAC}^0_3$.*

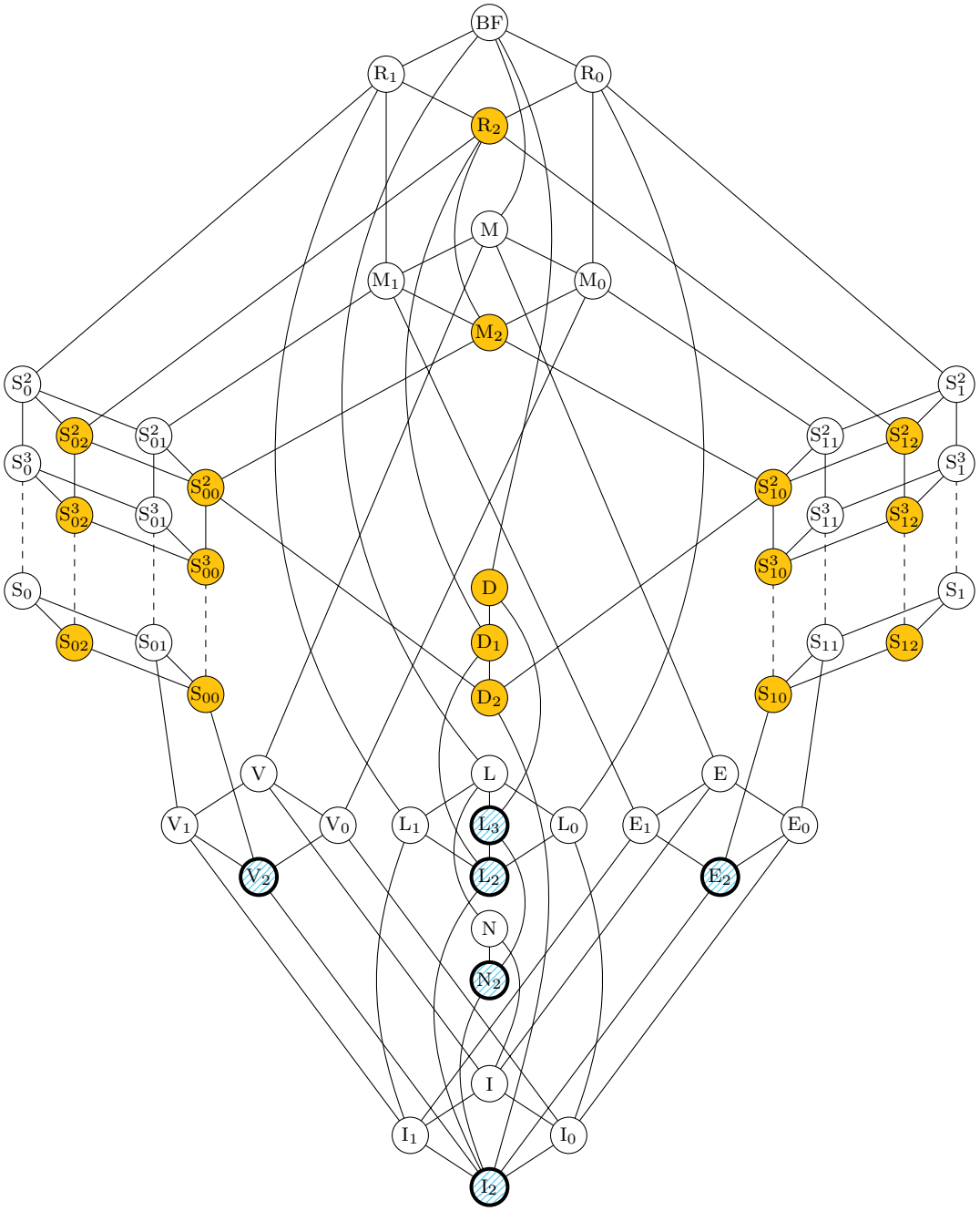


Figure 4.3: Illustration of Theorem 4.4.13. The vertices are colored with the *monotone circuit depth* complexity of deciding CSPs whose set of polymorphisms corresponds to the label of the vertex.

Proof. We write the proof in the case $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$. The other case is analogous.

From Lemmas 4.4.14 and 4.4.15 and Items (iv) and (v) of Lemma 4.4.3, we get that there is a monotone OR-reduction from $\mathsf{CSP\text{-}SAT}_S$ to $\mathsf{CSP\text{-}SAT}_{\{\mathsf{OR}^k, x, \neg x\}}$ for some k. However, an $\{\mathsf{OR}^k, x, \neg x\}$-formula is unsatisfiable iff there exists a literal and its negation as a constraint in the formula, or if there exists a disjunction in the formula such that every one of its literals appears negatively as a constraint. This condition can be easily checked by a polynomial-size monotone DNF. Composing the monotone DNF with the monotone OR-reduction, we obtain a depth-3 AC^0 circuit computing $\mathsf{CSP\text{-}SAT}_S$. □

Lemma 4.4.17 ([ABI+09, Lemma 3.8])**.** *Let S be a finite set of relations such that $\mathsf{Pol}(S) \subseteq \mathrm{R}_2$. If $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S)$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S)$, and S can express equality, then $\mathsf{CSP\text{-}SAT}_S$ is L-hard under $\leqslant_m^{\mathsf{AC}^0}$ reductions.*

Lemma 4.4.18. *Let S be a finite set of relations. If $\mathrm{S}_{02} \not\subseteq \mathsf{Pol}(S)$ and $\mathrm{S}_{12} \not\subseteq \mathsf{Pol}(S)$, then $\mathsf{CSP\text{-}SAT}_S$ is L-hard or trivial.*

Proof. This follows by inspecting Post's lattice (Figure 4.1) and the classification theorem (Theorem 4.4.7). □

We may now prove the main result of this subsection.

Theorem 4.4.19. *We have $\mathsf{CSP} \cap \mathsf{AC}^0 \subseteq \mathsf{mAC}^0_3$. Moreover, if $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mAC}^0_3$, then $\mathsf{CSP\text{-}SAT}_S$ is L-hard under $\leqslant_m^{\mathsf{AC}^0}$ reductions.*

Proof. Let S be a finite set of relations. If $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mAC}^0_3$, then, by Lemma 4.4.16, at least one of the following cases hold:

1. $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$, and S can express the equality relation;
2. $\mathrm{S}_{02} \not\subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ and $\mathrm{S}_{12} \not\subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$.
3. $\mathsf{Pol}(S) \not\subseteq \mathrm{R}_2$.

Since $\mathsf{CSP\text{-}SAT}_S$ is not trivial, we obtain that $\mathsf{CSP\text{-}SAT}_S$ is L-hard in the first two cases by Lemmas 4.4.17 and 4.4.18, and it's easy to check that $\mathsf{CSP\text{-}SAT}_S$ is also L-hard in the third case by inspecting Post's lattice (Figure 4.1) and the classification theorem (Theorem 4.4.7). Since $\mathsf{L} \not\subseteq \mathsf{AC}^0$, this also implies that, if $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0$, then $\mathrm{S}_{02} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$ or $\mathrm{S}_{12} \subseteq \mathsf{Pol}(S) \subseteq \mathrm{R}_2$, and S cannot express the equality relation. Lemma 4.4.16 again gives $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mAC}^0_3$. □

4.4.5 Consequences for monotone circuit lower bounds via lifting

We now prove a stronger form of Theorem 1.3.4. In the previous section, we showed that $\mathsf{CSP} \cap \mathsf{AC}^0 \subseteq \mathsf{mAC}^0$. In particular, this means that there does not exist a finite set of relations S such that $\mathsf{CSP\text{-}SAT}_S$ separates AC^0 and mNC^1, a separation which we proved in Theorem 1.3.1. We will also observe that, if $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^2$, then $\mathsf{CSP\text{-}SAT}_S$ is $\oplus\mathsf{L}$-hard.

Theorem 4.4.20. *Let S be a finite set of Boolean relations.*

1. *If $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mAC}^0_3$ then $\mathsf{CSP\text{-}SAT}_S$ is L-hard under $\leqslant_m^{\mathsf{AC}^0}$ reductions.*
2. *If $\mathsf{CSP\text{-}SAT}_S \notin \mathsf{mNC}^2$, then $\mathsf{CSP\text{-}SAT}_S$ is $\oplus\mathsf{L}$-hard under $\leqslant_m^{\mathsf{AC}^0}$ reductions.*

Proof. Item (1) follows from Theorem 4.4.19. To prove item (2), suppose that $\mathsf{mDEPTH}(\mathsf{CSP\text{-}SAT}_S) = \omega(\log^2 n)$. Then, by Theorem 4.4.13, we conclude that $\mathsf{Pol}(S) \subseteq \mathrm{L}_3$, or $\mathsf{Pol}(S) \subseteq \mathrm{V}_2$, or $\mathsf{Pol}(S) \subseteq \mathrm{E}_2$. Theorem 4.4.7 implies that $\mathsf{CSP\text{-}SAT}_S$ is $\oplus\mathsf{L}$-hard. □

Further Discussion. We recall the discussion of Section 1.3.3. We introduced and defined the functions $\mathsf{CSP\text{-}SAT}_S$ in that section, as a way to capture monotone circuit lower bounds proved via lifting. This in particular captures the monotone function 3-XOR-SAT, which was proved in [GKRS19] to require monotone circuit lower bounds of size $2^{n^{\Omega(1)}}$ to compute, even though $\oplus\mathsf{L}$-machines running in polynomial-time can compute it. Theorem 4.4.20 proves that this separation between monotone and non-monotone circuit lower bounds cannot be improved by varying the set of relations S, as we argue below.

There are two ways one could try to find a function in AC^0 with large monotone complexity using a CSP-SAT function. First, one could try to define a set of relations S such that $\mathsf{CSP\text{-}SAT}_S \in \mathsf{AC}^0$, but the monotone complexity of $\mathsf{CSP\text{-}SAT}_S$ is large. However, Item (1) of Theorem 4.4.20 proves that this is impossible, as any CSP-SAT function outside of mAC^0 is L-hard under simple reductions and, therefore, cannot be computed in AC^0.

Secondly, one could try to be apply the arguments of Section 4.2, consisting of a padding trick and a simulation theorem. When S is the set of 3XOR relations, then indeed we obtain a function in $\mathsf{AC}^0[\oplus]$ with superpolynomial monotone circuit complexity, as proved in Theorem 4.2.1. However, Item (2) of Theorem 4.4.20 proves that this is best possible, as any CSP-SAT function which admits a superpolynomial monotone circuit lower bound must be $\oplus\mathsf{L}$-hard and, therefore, at least as hard as 3-XOR-SAT for non-monotone circuits. Item (2) also shows that even CSP-SAT

functions with a $\omega(\log^2 n)$ monotone depth lower bound must be $\oplus$L-hard, which suggests that the arguments of Section 4.2 applied to a CSP-SAT function are not able to prove the separation of Theorem 4.2.4.

A caveat to these impossibility results is in order. First, we only study Boolean-valued CSPs here, though the framework of lifting can also be applied in the context of non-Boolean CSPs. It's not clear if non-Boolean CSPs exhibit the same dichotomies for monotone computation we proved in this section. We remark that Schaefer's dichotomy for Boolean-valued CSPs [Sch78] has been extended to non-Boolean CSPs [Zhu17, Bul17].

Secondly, the instances of CSP-SAT generated by lifting do not cover the entirety of the minterms and maxterms of CSP-SAT. In particular, our results do not rule out the possibility that a clever interpolation of the instances generated by lifting may give rise to a function that is easier to compute by non-monotone circuits, and therefore bypasses the hardness results of Theorem 4.4.20. One example is the Tardós function [Tar88]. A lifting theorem applied to a Pigeonhole Principle formula can be used to prove a lower bound on the size of monotone circuits that accept cliques of size k and reject graphs that are $(k-1)$-colorable, for some $k = n^{\varepsilon}$ [RM99, Rez23]. A natural interpolation for these instances would be the Clique(n, k) function, which, being NP-complete, would be related to an NP-complete CSP-SAT. However, as proved by [Tar88], there is a monotone function in P which has the same output behaviour over these instances.

4.5 Schaefer's Theorem in Monotone Complexity

We now give the deferred proofs of lemmas necessary for the results of Section 4.4. We will verify that the reductions between Boolean CSP problems via polymorphisms, as presented in [BCRV03, BCRV04, ABI+09], can also be implemented in the monotone setting.

4.5.1 Connectivity and generation functions

We recall the definitions of two prominent monotone Boolean functions that have efficient monotone circuits. Let ST-CONN : $\{0,1\}^{n^2} \to \{0,1\}$ be the function that outputs 1 on a given directed graph G if there exists a path from 1 to n in G. Let GEN : $\{0,1\}^{n^3} \to \{0,1\}$ be the Boolean function which receives a set T of triples $(i,j,k) \in [n^3]$, and outputs 1 if $n \in S$, where $S \subseteq [n]$ is the set generated with the following rules:

- *Axiom:* $1 \in S$,
- *Generation:* If $i, j \in S$ and $(i, j, k) \in T$, then $k \in S$.

The following upper bounds are well-known and easy to prove.

Theorem 4.5.1 ([Juk12, Exercise 7.3], [RM99])**.** *We have* $\mathsf{ST\text{-}CONN} \in \mathsf{mNL}$ *and* $\mathsf{GEN} \in \mathsf{mSIZE}[\mathsf{poly}]$.

4.5.2 Proof of reduction lemmas

Here we present monotonised versions of the proofs of [BCRV04, Propositions 2.2 - 2.4], which give a simplified presentation of the results of [Sch78].

Lemma 4.4.14 ([BCRV04, Proposition 2.3])**.** *Let S_1 and S_2 be finite sets of relations. If $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, then there exists a constant $C \in \mathbb{N}$ such that $\mathsf{CSP\text{-}SAT}^n_{S_1} \leqslant^{\mathsf{mOR}}_m \mathsf{CSP\text{-}SAT}^{Cn}_{S_2}$.*

Proof. If $\mathsf{COQ}(S_1) \subseteq \mathsf{COQ}(S_2)$, then each relation of S_1 can be represented as a conjunctive query over S_2. Let F_1 be a S_1-formula. For each constraint C_1 of F_1, there exists a formula $\varphi(C_1)$ in $\mathsf{CNF}(S_2)$ such that C_1 is a projection of $\varphi(C_1)$ (i.e., C_1 is a conjunctive query of $\varphi(C_1)$). However, note that C_1 is satisfiable if and only if $\varphi(C_1)$ is satisfiable. So we can replace the constraint C_1 by the *set* of constraints in $\varphi(C_1)$. Doing this for every constraint in F_1, we obtain an S_2-formula F_2 which is satisfiable iff F_1 is satisfiable.

Now note that, to decide if a given constraint application C of S_2 is in the reduction, it suffices to check if there exists a S_1-constraint C_1 in F_1 such that C is in $\varphi(C_1)$. Using non-uniformity, this can be easily done by an OR over the relevant input bits.[7]

Finally, we observe that, since the arities of each relation in S_1 and S_2 are constant, we only add a constant number of variables for each constraint to represent S_1-formulas with conjunctive queries over S_2-formulas. □

Lemma 4.5.2. *Let S be a set of Boolean relations. We have* $\mathsf{CSP\text{-}SAT}_{S\cup\{=\}} \leqslant^{\mathsf{mNL}}_m \mathsf{CSP\text{-}SAT}_S$.

Proof. Let F be a $(S \cup \{=\})$-formula on n variables given as an input. Remember that F is given as a Boolean vector α, where each bit of α represents the presence of a constraint application on n variables from $S \cup \{=\}$. We first build an undirected

[7] Even without non-uniformity, one can 'brute-force' over all relations in S_1 and S_2 to find the right representation of a relation in S_1 as an S_2-formula. This is possible since S_1 and S_2 are finite.

graph G with the variables $x_1, \ldots, x_n$ as vertices, and we put an edge between x_i and x_j if the constraint $x_i = x_j$ appears in F. Note that G can be constructed by a monotone projection from F.

Let $R \in S$ and let $C = R(x_1, \ldots, x_n)$ be a constraint application of R. If C appears in F, we add to the system every constraint of the form $C' = R(y_1, \ldots, y_n)$ such that, for every $i \in [n]$, there exists a path from x_i to y_i in the graph G. In this case, we say that C *generates* C'. Let F_2 be the formula that contains all non-equality constraints of F, and all the non-equality constraints generated by a constraint in F. It's not hard to see that F is satisfiable if and only if F_2 is satisfiable, and therefore the reduction is correct.

Moreover, the reduction can be done in monotone NL using the monotone NL algorithm for ST-CONN (Theorem 4.5.1). Indeed, there are at most n^k constraint applications of a given relation R of arity k. Therefore, to decide if a constraint $C' = R(y_1, \ldots, y_n)$ appears in F_2, it suffices to check if there exists a constraint application of R in F which generates C'. This can be checked with n^k calls to ST-CONN. □

Lemma 4.4.4 (Polymorphisms characterise the complexity of CSPs [Jea98, BCRV04, Theorem 2.4])**.** *If* $\mathsf{Pol}(S_2) \subseteq \mathsf{Pol}(S_1)$*, then* $\mathsf{CSP\text{-}SAT}^n_{S_1} \leqslant^{\mathsf{mNL}}_m \mathsf{CSP\text{-}SAT}^{\mathsf{poly}(n)}_{S_2}$.

Proof. If $\mathsf{Pol}(S_2) \subseteq \mathsf{Pol}(S_1)$, then from Lemma 4.4.3 (Items (iii), (v) and (vi)) we obtain $\mathsf{COQ}(S_1) \subseteq \langle S_1 \rangle \subseteq \langle S_2 \rangle = \mathsf{COQ}(S_2 \cup \{=\})$. Therefore, by Lemmas 4.4.14 and 4.5.2 we can do the following chain of reductions in monotone NL:

$$\mathsf{CSP\text{-}SAT}_{S_1} \leqslant^{\mathsf{mOR}}_m \mathsf{CSP\text{-}SAT}_{S_2 \cup \{=\}} \leqslant^{\mathsf{mNL}}_m \mathsf{CSP\text{-}SAT}_{S_2}.$$ □

4.5.3 Monotone circuit upper bounds

We restate and prove the theorem.

Theorem 4.4.5 (Monotone version of the upper bounds for CSP-SAT [Sch78, ABI+09])**.** *Let S be a finite set of relations. The following holds.*

1. *If* $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$ *or* $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$*, then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mSIZE}[\mathsf{poly}]$.
2. *If* $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$*, or* $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$*, or* $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$*, then* $\mathsf{CSP\text{-}SAT}_S \in \mathsf{mNL}$.

Proof. We prove each case separately.

Proof of (1). We first observe that 3-Horn-SAT (see definition in Section 4.4.2) can be solved by a reduction to $\mathsf{GEN} \in \mathsf{mSIZE}[\mathsf{poly}]$. Indeed, we interpret each constraint of the form $(\neg x_i \vee \neg x_j \vee x_k)$ (which is equivalent to $x_i \wedge x_j \implies x_k$)

as a triple (i, j, k), and constraints of the form x_i as a triple $(0, 0, i)$. Let $S \subseteq \{0, 1, 2 \ldots, n\}$ be the set generated by these triples, applying the generation rules of GEN, using $0 \in S$ as the axiom. It suffices to check that there exists some constraint of the form $\neg x_i \vee \neg x_j \vee \neg x_k$, such that $\{i, j, k\} \subseteq S$[8]. This process can be done with polynomial-size monotone circuits, invoking GEN. Therefore, it follows from Theorem 4.5.1 that 3-Horn-SAT $\in$ mSIZE[poly].

Moreover, we recall that, if $\mathrm{E}_2 \subseteq \mathsf{Pol}(S)$, then $S \subseteq \mathsf{COQ}(\mathcal{H}^3)$ (in other words, every S-formula can be written as a set of 3-Horn equations – see Lemma 4.4.6). Therefore, from Items (iv) and (v) of Lemma 4.4.3 and Lemma 4.4.14, we conclude that $\mathsf{CSP\text{-}SAT}_S \leqslant_m^{\mathsf{mOR}}$ 3-Horn-SAT $\in$ mSIZE[poly].

Now recall that, if $\mathrm{V}_2 \subseteq \mathsf{Pol}(S)$, then $S \subseteq \mathsf{COQ}(\mathcal{A}^3)$, where $\mathcal{A}^3$ is the set of width-3 Anti-Horn relations (i.e., $\mathcal{A}^3 = \{(x_1 \vee x_2 \vee \neg x_3), (x_1 \vee x_2 \vee x_3), (\neg x)\}$; see Lemma 4.4.6). But note that an $\mathcal{A}^3$-formula φ is satisfiable if and only if the $\mathcal{H}^3$-formula $\varphi(\neg x_1, \ldots, \neg x_n)$ is satisfiable. Therefore by Lemma 4.4.14 and Items (iv) and (v) of Lemma 4.4.3, we have $\mathsf{CSP\text{-}SAT}_S \leqslant_m^{\mathsf{mOR}} \mathsf{CSP\text{-}SAT}_{\mathcal{A}^3} \leqslant_m^{\mathsf{mProj}}$ 3-Horn-SAT $\in$ mSIZE[poly].

Proof of (2). We first prove the case $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$. Recall 2-SAT $= \mathsf{CSP\text{-}SAT}_\Gamma$, where $\Gamma = \{(x_1 \vee x_2), (x_1 \vee \neg x_2), (\neg x_1 \vee \neg x_2)\}$. It's easy to check that the standard reduction from 2-SAT to ST-CONN can be done in monotone NL (see [JLL76, Theorem 4]). Therefore, it follows from Theorem 4.5.1 that 2-SAT $\in$ mNL. Now, recall that, if $\mathrm{D}_2 \subseteq \mathsf{Pol}(S)$, then $S \subseteq \mathsf{COQ}(\Gamma)$ (Lemma 4.4.6). Therefore, from Lemma 4.4.14 and Items (iv) and (v) of Lemma 4.4.3, we conclude $\mathsf{CSP\text{-}SAT}_S \in$ mNL.

We now suppose that $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$. We check that the proof of [ABI+09, Lemma 3.4] gives a monotone circuit. If $\mathrm{S}_{00} \subseteq \mathsf{Pol}(S)$, then there exists $k \geqslant 2$ such that ${\mathrm{S}_{00}}^k \subseteq \mathsf{Pol}(S)$ (that's because there does not exist a finite set of relations S such that $\mathsf{Pol}(S) = \mathrm{S}_{00}$). Note that ${\mathrm{S}_{00}}^k = \mathsf{Pol}(\Gamma)$, where $\Gamma = \{\mathsf{OR}^k, x, \neg x, \rightarrow, =\}$. We show below how to decide if a Γ-formula is unsatisfiable in monotone NL. The result then follows from Lemma 4.4.4.

Let F be a given Γ-formula with n variables. We first construct a directed graph G, with vertex set $\{x_1, \ldots, x_n\}$, and with arcs (x_i, x_j) if $x_i \rightarrow x_j$ is a constraint of F, and arcs (x_i, x_j) and (x_j, x_i) if $x_i = x_j$ is a constraint of F. This can be done with a monotone projection. Observe that a Γ-formula F is unsatisfiable if, and only if, there exists a constraint of the form $x_{i_1} \vee \cdots \vee x_{i_k}$ in F, such that there exists a path from some x_{i_j} to a constraint $\neg y$ in F. This can be checked in monotone NL

[8] The constraint $\neg x_i \vee \neg x_j \vee \neg x_k$ corresponds to the statement that $i \notin S$ *or* $j \notin S$ *or* $k \notin S$. This statement is *false* when $\{i, j, k\} \subseteq S$ – thus, we find a contradiction with GEN when all these three (not necessarily distinct) elements are in S.

by Theorem 4.5.1.

The case $\mathrm{S}_{10} \subseteq \mathsf{Pol}(S)$ is analogous. □

Chapter 5

On the Computational Hardness of Quantum One-Wayness

Abstract

There is a large body of work studying what forms of computational hardness are needed to realize classical cryptography. In particular, one-way functions and pseudorandom generators can be built from each other, and thus require equivalent computational assumptions in order to be realized. Furthermore, the existence of either of these primitives implies that $\mathsf{P} \neq \mathsf{NP}$, which gives a lower bound on the necessary hardness.

One can also define versions of each of these primitives with quantum output: respectively one-way state generators and pseudorandom state generators. Unlike in the classical setting, it is not known whether either primitive can be built from the other. Although it has been shown that pseudorandom state generators for certain parameter regimes can be used to build one-way state generators, the implication has not been previously known in full generality. Furthermore, to the best of our knowledge the existence of one-way state generators has no known implications in traditional complexity theory.

We show that pseudorandom states compressing n bits to $\log n + 1$ qubits can be used to build one-way state generators and that pseudorandom states compressing n bits to $\omega(\log n)$ qubits are *themselves* one-way state generators. This is a nearly optimal result, since pseudorandom states with fewer than $c \log n$-qubit output can be shown to exist unconditionally. We also show that any one-way state generator can be broken by a quantum algorithm with classical access to a PP oracle.

An interesting implication of our results is that a $t(n)$-copy one-way state generator exists unconditionally, for every $t(n) = o(n/\log n)$. This contrasts nicely with the previously known fact that $O(n)$-copy one-way state generators require computational hardness. We also outline a new route towards a black-box separation between one-way state generators and quantum bit commitments.

Organisation of the chapter

We begin with a review of results and definitions (Section 5.1), and then move forward in Section 5.2 to show how one can obtain OWSGs from PRSs even when the PRS is 'compressing'. In that same section, we will also show how to obtain fixed-copy OWSGs. Our final technical section will show how to break OWSGs with a PP oracle (Section 5.3).

5.1 Preliminaries

In this section, we introduce basic notation and recall definitions from the literature that will be used throughout the rest of the chapter.

5.1.1 Basic quantum computing

Throughout this chapter, we will refer to an *n-qubit pure state* as a unit vector in $\mathbb{C}^{2^n}$. We will follow the 'bra-ket' notation, and write a unit vector $\phi \in \mathbb{C}^{2^n}$ as $|\phi\rangle$, and ϕ^* as $\langle\phi|$. We will equivalently denote by $\mathbb{S}(N)$ the set of N-dimensional pure quantum states. We identify the set of n-qubit pure states with $\mathbb{S}(2^n)$.

We will assume familiarity with the notion of *quantum circuits*, for which good references are [AB09, Section 10.3] or [NC16, Section 1.3.4]. For this work, it suffices to think of quantum circuits as a sequence of applications of unitary matrices of constant size to the qubits of the system. Each of these unitary matrices is taken from a fixed finite set, called the *gate set*. By the Solovay-Kitaev theorem (see [AB09, Theorem 10.12] or [NC16, Appendix 3]), the $\{\mathsf{CNOT}, H, T\}$-gate set is enough to efficiently simulate quantum circuits over a different gate set with a desired accuracy. In particular, all the algorithms, complexity classes and cryptographic primitives we study in this chapter will be robust with respective to the choice of the gate set, unless otherwise noted.[1]

We define a quantum polynomial-time algorithm (QPT) to be a uniformly generated sequence of polynomial-size quantum circuits. In this work, a quantum algorithm may either have *classical (binary)* output or *quantum* output (i.e., a pure quantum state). When the algorithm has classical output, a fixed set of the qubits is fixed beforehand, and the output of the computation is defined as the result of measuring those qubits after applying the circuit. In particular, we may assume without loss of generality that a quantum circuit for a decision problem $L \subseteq \{0,1\}^*$ outputs

[1] The only exception is the complexity class PostBQP and its promise-variant PromisePostBQP, which we will discuss in the next subsection.

the measurement of the first qubit. In the next section, when discussing quantum algorithms with *postselection*, it will be useful to consider quantum algorithms with output in $\{0,1,*\}$ – this can be done by measuring the first 2 qubits, and if the first qubit measures to 0 we output $*$, otherwise we output the measurement of the second qubit.

We will also consider quantum algorithms with *quantum* output. We say that a quantum circuit G maps n bits to a m-qubit state if, for all $k \in \{0,1\}^n$, we have $G(|k\rangle|0\dots0\rangle) = |\phi_k\rangle_A \otimes |\nu_k\rangle_B$, where $|\phi_k\rangle$ is a m-qubit state. In other words, the circuit G receives an input k and some 'workspace qubits' $|0\dots0\rangle$ (also known as 'ancilla' qubits) and generates $|\phi_k\rangle$ with a trash state $|\nu_k\rangle$ in the other part of the system, unentangled with $|\phi_k\rangle$. Formally, the state $|\phi_k\rangle$ can be obtained by applying the partial trace operator Tr_B on the output state $|\phi_k\rangle_A \otimes |\nu_k\rangle_B$.[2] To ease notation, we will simply write $G(k) = |\phi_k\rangle$ in the rest of this chapter to denote that G generates the state $|\phi_k\rangle$ when given $|k\rangle\,|0\dots0\rangle$, as just discussed. Moreover, the algorithm is said to be *polynomial-time* if the size of the circuit is polynomial in n – this means that the ancilla state $|0\dots0\rangle$ must have at most $\mathsf{poly}(n)$ qubits.

We will not consider non-uniform models in this chapter.

5.1.2 Computational complexity

We refer the reader to [AB09] for the definition of standard complexity classes such as BQP (Definition 10.9 of [AB09]) and PP (Definition 17.6 of [AB09]).

We recall the definition of the PostBQP complexity class (see, e.g., [Kre21, Definition 12]). We will only be concerned with its promise version in this chapter.

Definition 5.1.1 (PromisePostBQP)**.** *A promise problem $\Pi : \{0,1\}^* \to \{0,1,\perp\}$ is in* PromisePostBQP *(Postselected Bounded-error Quantum Polynomial time) if there exists a QPT algorithm $\mathcal{A}$ whose output is in $\{0,1,*\}$, and which is such that:*

(i) For all $x \in \{0,1\}^*$, we have $\mathbb{P}[\mathcal{A}(x) \in \{0,1\}] > 0$. When $\mathcal{A}(x) \in \{0,1\}$, we say that postselection succeeds*.*

(ii) If $\Pi(x) = 1$, then $\mathbb{P}[\mathcal{A}(x) = 1 \mid \mathcal{A}(x) \in \{0,1\}] \geqslant \frac{2}{3}$. In other words, conditioned on postselection succeeding, $\mathcal{A}$ outputs 1 with at least $\frac{2}{3}$ probability.

(iii) If $\Pi(x) = 0$, then $\mathbb{P}[\mathcal{A}(x) = 0 \mid \mathcal{A}(x) \in \{0,1\}] \geqslant \frac{2}{3}$. In other words, conditioned on postselection succeeding, $\mathcal{A}$ outputs 1 with at most $\frac{1}{3}$ probability.

[2] For more on the partial trace and on discarding qubits of a bipartite state, see [NC16, Section 2.4.3].

We remark that the definition of PostBQP is sensitive to the choice of the gate-set, as noticed by [Kup15]. We remark that [Kup15] also proves that every gate-set that satisfies two "reasonable" conditions gives rise to an equivalent "canonical" PostBQP class. Throughout the chapter we assume that we are dealing with one such gate-set, such as the $\{\mathsf{CNOT}, H, T\}$-gate-set. We refer the reader to [Kup15, Section 2.5] for a detailed technical discussion of this matter.

We also recall a result of Aaronson [Aar05], which states that PP = PostBQP, remarking that his result also holds for the corresponding promise classes. Let us first define PromisePP.

Definition 5.1.2. *A promise problem* $\Pi : \{0,1\}^* \to \{0,1,\bot\}$ *is in* PromisePP *if there exists a randomized polynomial-time classical algorithm* $\mathcal{A}$ *such that:*

(i) If $\Pi(x) = 1$*, then* $\mathbb{P}[\mathcal{A}(x) = 1] > \frac{1}{2}$.

(ii) If $\Pi(x) = 0$*, then* $\mathbb{P}[\mathcal{A}(x) = 1] \leqslant \frac{1}{2}$.

Lemma 5.1.3 (Aaronson [Aar05])**.** PromisePostBQP = PromisePP.

Since PP is a syntactic class, we can extend any promise problem in PromisePP into a language in PP. This remark will be fundamental in Section 5.3, as it was also in [Kre21], when we build a PP oracle with which we can break OWSGs.

5.1.3 Quantum information theory and cryptography

We say that a function $\alpha : \mathbb{N} \to \mathbb{N}$ satisfies $\alpha \in \mathsf{negl}(n)$ if the asymptotic growth of α is $\alpha(n) = n^{-\omega(1)}$. We will often simply write $\mathsf{negl}(n)$ in equations and inequalities to denote a function in $\mathsf{negl}(n)$.

The *Haar measure* over $\mathbb{S}(2^n)$, which we denote by $\mathrm{Haar}(n)$, is defined as the unique unitarily invariant measure over $\mathbb{S}(2^n)$. This means that, for every unitary U, if a random state $|\phi\rangle$ is distributed according to the Haar measure, then $U|\phi\rangle$ also is. A proof of the uniqueness of this measure can be found in [Wat18, Corollary 7.23], where it is referred to as the *uniform spherical measure*[3].

An important alternative definition of the Haar measure is given by the following lemma, which seems to have been first observed by Muller [Mul59]. We assume familiarity with the *Gaussian distribution*.

[3]Following the current literature on quantum cryptography (e.g., [AGQY22, Kre21]), we decide to call it the 'Haar measure' instead of the 'uniform spherical measure'. Quantum cryptographers call it the Haar measure on states because the *Haar measure on unitaries* – corresponding to a notion of a uniformly random unitary matrix, though we will not discuss it in this work – induces the Haar measure on states in the following way. Fix any initial state $|\phi\rangle$ (e.g., the $|0\dots0\rangle$ state), and sample a Haar-random unitary U. It is known that the distribution of pure states given by $U|\phi\rangle$ follows the Haar distribution ([Wat18, Theorem 7.22]).

Lemma 5.1.4 ([Wat18, Corollary 7.23]). *Let $n \in \mathbb{N}$. Sampling from the Haar distribution* $\mathrm{Haar}(n)$ *is equivalent to sampling $|\psi\rangle$ as*

$$|\psi\rangle \leftarrow \frac{1}{\sqrt{\sum_{x\in\{0,1\}^n} \alpha_x^2 + \beta_x^2}} \cdot \sum_{x\in\{0,1\}^n} (\alpha_x + \beta_x i)\,|x\rangle\,,$$

where each α_x, β_x is sampled according to the standard Gaussian distribution with expectation 0 and standard deviation 1.

Quantum cryptography. We start by introducing the notion of pseudorandom state generators (PRS) [JLS18]. Roughly speaking, given a classical key k, a PRS maps k to a quantum pure state $|\phi_k\rangle$. The security guarantee is that the output of a PRS on a random input should look like a random state. That is, it is hard for any quantum adversary to distinguish a random $|\phi_k\rangle$ from a Haar random state.

Definition 5.1.5 (Pseudorandom States, Definition 2 of [JLS18]). *Let λ be the security parameter and let $\mathcal{K} = \mathcal{K}(\lambda)$ be a set of binary strings referred to as the* key space*, with the property that every* key *$k \in \mathcal{K}$ has at most $\mathsf{poly}(\lambda)$ bits. Let G be a QPT algorithm that on input $k \in \mathcal{K}$ outputs a pure quantum state $|\phi_k\rangle$ over $n = n(\lambda)$ qubits. We say G is (t, ε)-pseudorandom if the distribution over outputs is ε-indistinguishable from Haar random given t copies. In other words, for any QPT adversary $\mathcal{A}$, we have*

$$\left| \mathop{\mathbb{P}}_{k\leftarrow\mathcal{K}}[\mathcal{A}(|\phi_k\rangle^{\otimes t}) = 1] - \mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(n)}[\mathcal{A}(|\psi\rangle^{\otimes t}) = 1] \right| \leqslant \varepsilon.$$

We say that G is pseudorandom if it is $\left(\lambda^c, \frac{1}{\lambda^c}\right)$-pseudorandom for all $c > 0$, and that it is t-pseudorandom or a t-copy PRS if it is $\left(t, \frac{1}{\lambda^c}\right)$-pseudorandom for all $c > 0$.

We turn to the notion of one-way state generators (OWSGs) [MY22a]. A OWSG maps a classical key k to quantum state $|\phi_k\rangle$. The security guarantee of a OWSG is that, given any polynomial number of copies of $|\phi_k\rangle$, it is hard for a quantum algorithm to find keys k' such $|\phi_k\rangle, |\phi_{k'}\rangle$ have noticeable overlap. OWSGs can also be defined to have mixed state outputs [MY22b], although we will not consider this variant in this work.

Definition 5.1.6 (One-Way State Generators, Definition 4.1 of [MY22b]). *Let λ be the security parameter and let $\mathcal{K} = \mathcal{K}(\lambda)$ be a set of binary strings referred to as the* key space*, with the property that every* key *$k \in \mathcal{K}$ has at most $\mathsf{poly}(\lambda)$ bits. Let G be a QPT algorithm that on input $k \in \mathcal{K}$ outputs a pure quantum state $|\phi_k\rangle$. We*

say G is (t,ε)-one-way if the outputs are hard to invert with accuracy at least ε. In other words, for any QPT adversary $\mathcal{A}$, we have

$$\mathop{\mathbb{E}}_{\substack{k\leftarrow\mathcal{K}\\ k'\leftarrow\mathcal{A}(|\phi_k\rangle^{\otimes t})}}\left[|\langle\phi_k|\phi_{k'}\rangle|^2\right]\leqslant\varepsilon.$$

We say that G is one-way or strongly one way if it is $\left(\lambda^c,\frac{1}{\lambda^c}\right)$-one-way for all $c>0$, and that it is t-one-way, a t-copy OWSG, or a t-copy strong OWSG if it is $\left(t,\frac{1}{\lambda^c}\right)$-one-way for all $c>0$.

We remark that all of the results in this chapter will also hold for the more general definition of (pure-state) one-way state generators that was introduced in the later work of Morimae and Yamakawa [MY22a], where the one-way state generator is allowed to have a separate quantum key generation procedure.[4]

We will rely on the notion of one-way puzzles, defined in [KT23]. A one-way puzzle is a pair of algorithms $(\mathsf{Samp},\mathsf{Ver})$ where Samp samples a key-puzzle pair (k,s) such that $\mathsf{Ver}(k,s)$ outputs 1 with overwhelming probability. Samp is required to be an efficient quantum algorithm, and Ver is allowed to be any arbitrary function. The security requirement is that given s, it is hard for an adversary to find a k' such that $\mathsf{Ver}(k',s)=1$.

Definition 5.1.7 (One-Way Puzzles [KT23])**.** *Let λ be the security parameter. A one-way puzzle is a pair of sampling and verification algorithms $(\mathsf{Samp},\mathsf{Ver})$ with the following syntax.*

- $\mathsf{Samp}(1^\lambda)\to(k,s)$ *is a (uniform) quantum polynomial time algorithm that outputs a pair of classical strings (k,s). We refer to s as the puzzle and k as its key. Without loss of generality we may assume that $k\in\{0,1\}^\lambda$.*
- $\mathsf{Ver}(k,s)\to\top$ *or* $\bot$*, is an unbounded algorithm that on input any pair of classical strings (k,s) halts and outputs either $\top$ or $\bot$.*

These satisfy the following properties.

[4] The results shown in Section 5.2 imply the existence of one-way state generators in the sense of Definition 5.1.6 in various settings. Since these are more restricted objects than the ones considered in the more general definition of [MY22a], our results will also implies their existence in the same settings and with an analogous parameter regime. Furthermore, since our oracle QPT algorithm in Section 5.3 comes from an algorithm breaking *one-way puzzles*, and one-way puzzles were proved in [KT23] to follow from the "general" pure state OWSGs of [MY22a], our algorithm will therefore also break OWSGs with a quantum key generation procedure.

- ***Correctness.*** *Outputs of the sampler pass verification with overwhelming probability, i.e.,*

$$\mathop{\mathbb{P}}_{(k,s)\leftarrow\mathsf{Samp}(1^\lambda)}[\mathsf{Ver}(k,s)=\top]=1-\mathsf{negl}(\lambda).$$

- ***Security.*** *Given s, it is (quantumly) computationally infeasible to find k satisfying $\mathsf{Ver}(k,s)=\top$, i.e., for every polynomial-sized quantum circuit $\mathcal{A}$,*

$$\mathop{\mathbb{P}}_{(k,s)\leftarrow\mathsf{Samp}(1^\lambda)}[\mathsf{Ver}(\mathcal{A}(s),s)=\top]=\mathsf{negl}(\lambda).$$

An important recent result due to Khurana and Tomer [KT23] states that one-way state generators imply one-way puzzles. This will be crucial for our algorithm in Section 5.3.

Theorem 5.1.8 ([KT23, Theorem 4.2])**.** *If there exists a $(O(n),\mathsf{negl}(n))$-OWSG, then there exists a one-way puzzle.*

Remark 5.1.9 (A remark about the relationship between quantum-output primitives and post-quantum primitives)**.** *We note that post-quantum one-way functions (i.e., classical one-way functions that are secure against quantum adversaries) imply the existence of PRSs with any desired output length [BS20]. Since post-quantum pseudorandom generators (classical pseudorandom generators which are secure against quantum adversaries) trivally imply the existence of post-quantum one-way functions, one also observes that post-quantum pseudorandom generators imply the existence of pseudorandom states.*

We furthermore observe that, since post-quantum one-way functions imply the existence of PRSs of any desired length [BS20], and expanding PRSs imply OWSGs [MY22b], we can conclude that post-quantum one-way functions also imply the existence of one-way state generators. Moreover, as noted in Section 1.4.4, the reduction of [BS20] implies that our proof that PRSs are *OWSGs (Theorem 5.2.4) is optimal in its parameters.*

5.1.4 Probability distributions

We recall a few standard probability distributions, the chi-squared distribution and the Fisher-Snedecor distribution, the latter of which will be useful when trying to show that compressing PRSs are one-way state generators (Lemma 5.2.1).

Definition 5.1.10 (Chi-squared distribution)**.** *Let $X_1,\ldots,X_k$ be independent normal random variables with mean* 0 *and variance* 1*. The* chi-square distribution with

k degrees of freedom, *denoted by* $\chi^2(k)$, *is the probability distribution of the sum* $\sum_{i=1}^{k} X_i^2$.

Definition 5.1.11 (Fisher-Snedecor distribution)**.** *Let* $a, b \in \mathbb{N}$. *The* Fisher-Snedecor distribution with a and b degrees of freedom, *denoted by* $F(a,b)$, *is the distribution given by the following ratio:*

$$F(a,b) \sim \frac{\chi^2(a)/a}{\chi^2(b)/b} = \frac{b}{a} \cdot \frac{\chi^2(a)}{\chi^2(b)}.$$

We now recall the definition of the beta function and its variations in order to express the cumulative distribution function of the F-distribution. The *beta function* is defined as

$$B(a,b) := \int_0^1 x^{a-1}(1-x)^{b-1}\,dx.$$

The *incomplete beta function* is defined as

$$B(t;a,b) := \int_0^t x^{a-1}(1-x)^{b-1}\,dx.$$

The *regularised incomplete beta function* is defined as

$$I_x(a,b) := \frac{B(x;a,b)}{B(a,b)}.$$

We can now express the cumulative distribution function of the F-distribution. An exposition of this fact can be found in the encyclopedia entry [Cab11] or in [JKB95, Equation 27.8].

Lemma 5.1.12 (Cumulative distribution function of the F-distribution)**.** *The cumulative distribution function* $p_{a,b}(t)$ *of the* $F(a,b)$ *distribution satisfies*

$$p_{a,b}(t) = I_{at/(at+b)}(a/2, b/2).$$

In particular, we have

$$\mathop{\mathbb{P}}_{\mathbf{Y} \leftarrow F(a,b)}[\mathbf{Y} \leqslant \theta] = \frac{\int_0^{(a\theta)/(a\theta+b)} x^{a-1}(1-x)^{b-1}\,dx}{\int_0^1 x^{a-1}(1-x)^{b-1}\,dx}.$$

Given two probability distributions $\mathcal{D}_0, \mathcal{D}_1$ supported in a finite set X, we define their *total variation distance* $\mathsf{d}_{\mathsf{TV}}(\mathcal{D}_0, \mathcal{D}_1)$ as follows:

$$\mathsf{d}_{\mathsf{TV}}(\mathcal{D}_0, \mathcal{D}_1) := \frac{1}{2} \sum_{x \in X} \left| \mathcal{D}_0(x) - \mathcal{D}_1(x) \right|,$$

where $\mathcal{D}_i(x)$ is the probability of x being sampled by distribution $\mathcal{D}_i$ for $i \in \{0,1\}$.

5.1.5 Approximate t-designs

In our construction of unconditional t-copy OWSGs, we will use approximate t-designs. Informally, an approximate t-design is a distribution over states such that t copies of an output state is statistically close to t copies of a Haar random state.

We first recall that the *Trace norm* (also known as *Schatten 1-norm*) of a matrix M is defined as $\|M\|_1 := \mathsf{Tr}(\sqrt{M^*M})$.

Definition 5.1.13 (Approximate t-Design, Definition 2.2 of [OSP23], rephrased[5])**.** *A probability distribution S over $\mathbb{S}(2^n)$ is an ε-approximate t-design if*

$$\left\| \mathop{\mathbb{E}}_{|\psi\rangle \leftarrow \mathrm{Haar}(n)} \left[|\psi\rangle\langle\psi|^{\otimes t} \right] - \mathop{\mathbb{E}}_{|\psi\rangle \leftarrow S} \left[|\psi\rangle\langle\psi|^{\otimes t} \right] \right\|_1 \leqslant \varepsilon, \tag{5.1}$$

where $\|\cdot\|_1$ is the trace norm. We call G an efficient ε-approximate t-design *if G is a quantum algorithm running in time $\mathsf{poly}(n, m, t, \log(1/\varepsilon))$ which maps classical strings in $\{0,1\}^n$ to quantum states over $\mathbb{S}(2^m)$, and such that the output distribution of $G(\cdot)$ on a random n-bit string forms an ε-approximate t-design.*

Inequality (5.1) implies that no measurement can distinguish t copies of an ε-approximate t-design from t copies of a Haar random state with advantage greater than $1/2 + \varepsilon/4$[6]. In particular, no QPT algorithm can make this distinguishment, and thus t-designs are t-copy PRSs.

Recently, it has been shown that approximate t-designs with almost optimal seed exist unconditionally.

Theorem 5.1.14 (Theorem 1.1 of [OSP23], rephrased)**.** *For all $m, t, \varepsilon > 0$, there exists an efficient ε-approximate t-design with input size $n = O(mt + \log(1/\varepsilon))$.*

5.2 One-way state generators from compressing pseudorandom states

We construct both weak and strong t-copy one-way state generators from $(t+1)$-copy compressing pseudorandom states. Before this the only known reduction worked for expanding PRS mapping n-bit strings to cn-qubit states [MY22a] for

[5] The reference [OSP23] concentrates on *unitary* designs, a notion we will not consider in this work. However, unitary t-designs are known to imply our notion of (state) t-designs (see [Kre21, Proposition 18]).

[6] This is known as the Holevo-Helstrom theorem; see [Wat18, Theorem 3.4].

some $c > 1$. To generalise this reduction, we rely on the following concentration inequality, which informally states that, with high probability, any fixed state is unlikely to be close to Haar-random states. We will then use this in Theorem 5.2.2 to bound how well a OWSG inverter can distinguish outputs of a PRS from Haar-random states.

Lemma 5.2.1 (Concentration of Haar States[7])**.** *Let $|\phi_0\rangle$ be any state of dimension $N = 2^n$. Then, for any $s > 0$, we have*

$$\mathop{\mathbb{P}}_{|\psi\rangle\sim\mathrm{Haar}(n)}\left[|\langle\phi_0|\psi\rangle|^2 \geqslant \frac{1}{s}\right] \leqslant \left(\frac{s}{s+1}\right)^{N-1}.$$

Proof. Since the Haar distribution is invariant under unitary transformations, without loss of generality we assume $|\phi_0\rangle = |0\rangle$.

Recall (Lemma 5.1.4) that sampling from the Haar distribution is equivalent to sampling $|\psi\rangle$ as

$$\frac{1}{\sqrt{\sum_{x\in\{0,1\}^n} \alpha_x^2 + \beta_x^2}} \cdot \sum_{x\in\{0,1\}^n} (\alpha_x + \beta_x i)\,|x\rangle\,,$$

where each α_x, β_x is sampled according to the standard Gaussian with expectation 0 and standard deviation 1. Then, we define the random variable Y as

$$Y := \frac{\alpha_0^2 + \beta_0^2}{\sum_{x\neq 0} \alpha_x^2 + \beta_x^2}.$$

Expanding out the inner product gives us

$$|\langle 0|\psi\rangle|^2 = \frac{\alpha_0^2 + \beta_0^2}{\sum_x (\alpha_x^2 + \beta_x^2)} \leqslant Y.$$

But observe that each α_x, β_x is sampled independently, and so Y is distributed as the ratio of two chi-squared random variables. So, we see that Y is sampled as a (scaled) F-distribution as follows:

$$Y \sim \frac{\chi^2(2)}{\chi^2(2N-2)} \sim \frac{2}{2N-2} \cdot F(2, 2N-2).$$

[7] In [Kre21] the author refers to a closely related inequality as following from "standard concentration inequalities, or even an explicit computation". However we were unable to find an actual proof of the inequality in either the citation listed [BHH16], or in the paper which it cites [HLW06]. Consequently we decided to include a proof here for completeness.

We conclude by Lemma 5.1.12 that

$$\begin{aligned}
\mathbb{P}\left[Y \geqslant \frac{1}{s}\right] &= \mathbb{P}\left[F(2, 2N-2) \geqslant \frac{N-1}{s}\right] \\
&= 1 - \frac{\int_0^{\frac{1}{s+1}} (1-x)^{N-2} dx}{\int_0^1 (1-x)^{N-2} dx} \\
&= 1 - \frac{\left[-\frac{1}{N-1} \cdot (1-x)^{N-1}\right]_0^{1/(s+1)}}{\left[-\frac{1}{N-1} \cdot (1-x)^{N-1}\right]_0^1} \\
&= 1 - \frac{\left[(1-x)^{N-1}\right]_0^{1/(s+1)}}{\left[(1-x)^{N-1}\right]_0^1} \\
&= 1 + \left(\left(1 - \frac{1}{s+1}\right)^{N-1} - 1\right) \\
&= \left(\frac{s}{s+1}\right)^{N-1},
\end{aligned}$$

where $[f(x)]_0^1 = f(1) - f(0)$. Since

$$\mathop{\mathbb{P}}_{|\psi\rangle \sim \mathrm{Haar}(N)}\left[|\langle \phi_0 | \psi \rangle|^2 \geqslant \frac{1}{s}\right] \leqslant \mathbb{P}\left[Y \geqslant \frac{1}{s}\right],$$

we are done. □

Using this lemma we first show a general result stating that state generators that are pseudorandom must also be one-way. We then apply the result in two different parameter regimes. While [Kre21] only claims that PRSs with $m = n$ output bits can be broken by PP, [AGQY22] calls out that the proof can be extended to the case when $m \geqslant \log n + c$. For this regime, we show that PRSs are weak OWSGs. We then show that slightly less compressing PRSs ($m = \omega(\log n)$) are strong OWSGs. In all these three results, the number of copies falls by 1 moving from a PRS to a OWSG. While we end up focusing on sublinear-copy PRSs in the next subsection, these reductions work in the default many-copy setting considered in most papers on quantum cryptographic primitives.

Lemma 5.2.2. *For all $f(n)$ and for*

$$\delta = 2^n \cdot \left(\frac{f(n)}{f(n)+1}\right)^{(2^m-1)} + \frac{1}{f(n)},$$

if $G : k \mapsto |\phi_k\rangle$ is a state generator taking n-bit strings to m-qubit pure states which

is $(t+1,\varepsilon)$*-pseudorandom, then it is also* $(t,\varepsilon+\delta)$*-one way.*

Proof. For the sake of contradiction, assume that there exists an adversary $\mathcal{A}$ that can succeed with probability larger than $\varepsilon+\delta$ in the t-copy one-wayness game (Definition 5.1.6). We will construct a new adversary $\mathcal{A}'$ for the pseudorandomness game as follows:

Algorithm 1 Adversary $\mathcal{A}'$ in the pseudorandomness game

Input: $|\psi\rangle^{\otimes(t+1)}$.
Output: 0 if $|\psi\rangle$ is pseudorandom, 1 if $|\psi\rangle$ is Haar random.

1: Run $\mathcal{A}$ on the first t copies and obtain $k' \leftarrow \mathcal{A}(|\psi\rangle^{\otimes t})$
2:
3: Measure the last copy of $|\psi\rangle$ in the basis $\{|\phi_{k'}\rangle\langle\phi_{k'}|, I-|\phi_{k'}\rangle\langle\phi_{k'}|\}$
4:
5: **return** 1 if the result is $|\phi_{k'}\rangle\langle\phi_{k'}|$, else **return** 0.

Observe that when $\mathcal{A}'$ is given a pseudorandom input, it outputs 1 with probability at least

$$\mathop{\mathbb{P}}_{k\leftarrow\mathcal{K}}\left[\mathcal{A}'\left(|\phi_k\rangle^{\otimes(t+1)}\right)=1\right]=\mathop{\mathbb{E}}_{k\leftarrow\mathcal{K}}\left[|\langle\phi_k|\phi_{k'}\rangle|^2\middle|k'\leftarrow\mathcal{A}\left(|\phi_k\rangle^{\otimes t}\right)\right]>\varepsilon+\delta,$$

where the key space $\mathcal{K}$ is equal to $\{0,1\}^n$ here. Thus, it suffices remains to bound the probability that $\mathcal{A}'$ detects a Haar random state:

$$\mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)}\left[\mathcal{A}'\left(|\psi\rangle^{\otimes(t+1)}\right)=1\right]\leqslant\delta.$$

By construction, we have

$$\begin{aligned}
&\mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)}\left[\mathcal{A}'\left(|\psi\rangle^{\otimes(t+1)}\right)=1\right]\\
&\qquad=\mathop{\mathbb{E}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)}\left[|\langle\psi|\phi_k\rangle|^2\middle|k\leftarrow\mathcal{A}\left(|\psi\rangle^{\otimes t}\right)\right]\\
&\qquad=\int_{\mathbb{S}(2^m)}d\mu(\psi)\cdot\sum_{k\in\{0,1\}^n}\mathbb{P}\left[k\leftarrow\mathcal{A}\left(|\psi\rangle^{\otimes t}\right)\right]\cdot|\langle\psi|\phi_k\rangle|^2\\
&\qquad\leqslant\int_{\mathbb{S}(2^m)}d\mu(\psi)\cdot\max_{k\in\{0,1\}^n}|\langle\psi|\phi_k\rangle|^2.
\end{aligned}$$

Where $\mu(\psi)$ is the Haar measure on the space of m-qubit pure states. We will

now partition the set of m-qubit pure states into the states that are 'close' to a pseudorandom state $|\phi_k\rangle$, and the states that are 'far' from all pseudorandom states. Formally, we define

$$A_f := \left\{ |\psi\rangle \in \mathbb{S}(2^m) \middle| \max_k |\langle\psi|\phi_k\rangle|^2 \geqslant \frac{1}{f(n)} \right\}.$$

The set of states that are 'far' from all pseudorandom states is its complement,

$$B_f := \left\{ |\psi\rangle \in \mathbb{S}(2^m) \middle| \max_k |\langle\psi|\phi_k\rangle|^2 < \frac{1}{f(n)} \right\}.$$

We proceed with computing the integral separately for the two sets:

$$\begin{aligned}
\int_{\mathbb{S}(2^m)} d\mu(\psi) \cdot \max_{k\in\{0,1\}^n} |\langle\psi|\phi_k\rangle|^2 &= \int_{A_f} d\mu(\psi) \cdot \max_{k\in\{0,1\}^n} |\langle\psi|\phi_k\rangle|^2 \\
&\quad + \int_{B_f} d\mu(\psi) \cdot \max_{k\in\{0,1\}^n} |\langle\psi|\phi_k\rangle|^2 \\
&\leqslant \int_{A_f} d\mu(\psi) + \max_{\substack{|\psi\rangle\in B_f \\ k\in\{0,1\}^n}} |\langle\psi|\phi_k\rangle|^2 \\
&< \mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)} \left[\exists k \middle| |\langle\psi|\phi_k\rangle|^2 \geqslant \frac{1}{f(n)} \right] + \frac{1}{f(n)} \\
&\leqslant \sum_{k\in\mathcal{K}} \mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)} \left[|\langle\psi|\phi_k\rangle|^2 \geqslant \frac{1}{f(n)} \right] + \frac{1}{f(n)}.
\end{aligned}$$

Lemma 5.2.1 implies that

$$\mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)} \left[\mathcal{A}'\left(|\psi\rangle^{\otimes(t+1)} \right) = 1 \right] \leqslant 2^n \cdot \left(\frac{f(n)}{1+f(n)} \right)^{2^m-1} + \frac{1}{f(n)} = \delta.$$

We conclude

$$\left| \mathop{\mathbb{P}}_{k\leftarrow\mathcal{K}} \left[\mathcal{A}'\left(|\psi\rangle^{\otimes(t+1)} \right) = 1 \right] - \mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)} \left[\mathcal{A}'\left(|\psi\rangle^{\otimes(t+1)} \right) = 1 \right] \right| > \varepsilon. \qquad \square$$

Using this general result we can specify to the two theorems below.

Theorem 5.2.3 (PRSs imply OWSGs)**.** *If G is a state generator taking n-bit strings to $m > \log n + c$ qubit states (with $c \geqslant 1$) which is $(t+1, \varepsilon)$-pseudorandom, then G is also $(t, \varepsilon + 3/4)$-one-way.*

Proof. Take Lemma 5.2.2 with $f(n) = 2$, and $m > \log n + c$. We get that

$$\begin{aligned}\delta &\leqslant 2^n \cdot \left(\frac{2}{3}\right)^{n \cdot 2^c - 1} + \frac{1}{2} \\ &= \frac{3}{2} \cdot \left(\frac{2^{2^c+1}}{3^{2^c}}\right)^n + \frac{1}{2}.\end{aligned}$$

Since $c > 1$, the left term becomes less than $1/4$ for sufficiently large n, and thus $\delta < 3/4$. □

Using amplification arguments from [MY22b], these $(t, \varepsilon + 3/4)$-OWSGs can be used to construct strong OWSGs with negligible probability of inversion.

This reduction is nearly optimal. [BCQ22] showed that there exist statistical many-copy PRSs with $c \log n$ output bits for some $c < 1$. As shown later, strong OWSGs must imply computational assumptions (e.g. $\mathsf{BQP} \neq \mathsf{PP}$) so they cannot be implied by statistical or information-theoretic primitives. Consequently there can be no proof that PRSs with output shorter than $c \log n$ qubits imply OWSGs, meaning our proof is optimal up to multiplicative constant factors.

We can also show that PRSs with superlogarithmic output are themselves strong OWSGs.

Theorem 5.2.4 (PRSs are strong OWSGs)**.** *If G is a state generator taking n-bit strings to $m = \omega(\log n)$-qubit states which is $(t+1, \varepsilon)$-pseudorandom, then it is also $(t, \varepsilon + \mathsf{negl}(n))$-one way.*

Proof. Take Lemma 5.2.2 and set $f(n) = \frac{2^m - 1}{2^{m/2}} - 1$. Note that $f(n) = 2^{\Omega(m)} = n^{\omega(1)}$. We get

$$\begin{aligned}\delta &= 2^n \cdot \left(\frac{f(n)}{f(n)+1}\right)^{2^m - 1} + \frac{1}{f(n)} \\ &\leqslant 2^n \cdot \left(1 - \frac{1}{f(n)+1}\right)^{2^m - 1} + \mathsf{negl}(n) \\ &\leqslant 2^n \cdot \left(\frac{1}{e}\right)^{2^{m/2}} + \mathsf{negl}(n) \\ &= 2^{n - \Omega(n^{\omega(1)})} + \mathsf{negl}(n) \\ &= \mathsf{negl}(n).\end{aligned}$$

□

5.2.1 Unconditional OWSGs from efficient approximate t-designs

We first observe that any efficient $2^{-\lambda}$-approximate t-design is also a t-copy PRS. This follows definitionally, since t copies from a t-design are statistically close

to t copies from a Haar random state.

Proposition 5.2.5. *Let G be an efficient $2^{-\lambda}$-approximate t-design that maps n-bit strings to m-qubit pure states. Denote $|\psi_k\rangle := G(k)$. Then for any QPT adversary $\mathcal{A}$,*

$$\left| \mathop{\mathbb{P}}_{k\leftarrow\{0,1\}^n} \left[\mathcal{A}\left(|\psi_k\rangle^{\otimes t}\right) = 1\right] - \mathop{\mathbb{P}}_{|\psi\rangle\leftarrow\mathrm{Haar}(m)} \left[\mathcal{A}\left(|\psi\rangle^{\otimes t}\right) = 1\right] \right| \leqslant \mathsf{negl}(\lambda).$$

In particular, the proposition holds definitionally for *any* (even inefficient) quantum algorithm $\mathcal{A}$, and so therefore must also hold for QPT adversaries. A simple corollary of Theorem 5.2.4 then shows that efficient approximate t-designs are also OWSGs.

Corollary 5.2.6. *Let G be an efficient ε-approximate t-design mapping n bits to $m = \omega(\log n)$ qubits. Then G is $(t, \varepsilon + \mathsf{negl}(n))$-one-way.*

Theorem 5.1.14 with $\varepsilon = 2^{-\lambda}$ gives that, for any polynomial t, $t(\lambda)$-copy PRSs exist unconditionally, and thus Theorem 5.2.4 concludes that $t(\lambda)$-copy OWSGs also exist unconditionally.

Contrast this with the recent result of Khurana and Tomer [KT23], where they show that $\Theta(n)$-copy OWSGs can be used to build quantum bit commitments (and thus require computational hardness [LC97]). This raises the question of what is the largest number of copies t (relative to the number of classical input bits n) for which OWSGs exist unconditionally. We show that the efficient approximate t-designs of [OSP23] approach this computational threshold up to a logarithmic factor.

Corollary 5.2.7. *For every function $\alpha = \alpha(n) = \omega(1)$, there exists a $\Theta\left(\frac{n}{\alpha \cdot \log n}\right)$-copy OWSG.*

Proof. From Theorem 5.1.14, we know that there exists some positive constant c such that, for $n = c \cdot mt + c\log(1/\varepsilon)$, there is an efficient ε-approximate t-design mapping n bits to m qubits.

Setting $\varepsilon = 2^{-\lambda}$, $n = 2c \cdot \lambda$, $t = \lambda/(\alpha \log n)$ and $m = \alpha \cdot \log n$ gives us

$$n = 2c\lambda = c\alpha t \log n + c\lambda = cmt + c\log(1/\varepsilon),$$

and thus we can build efficient ε-approximate $\Theta(n/(\alpha \log n))$-designs. But since $m = \omega(\log n)$, using such a design also gives a $\Theta\left(\frac{n}{\alpha \log n}\right)$-copy strong OWSG from Corollary 5.2.6. □

5.3 Breaking one-way state generators with a PP oracle

In this section, we show how to break one-way state generators (OWSGs) with a PP oracle. Since one-way state generators imply one-way puzzles (Theorem 5.1.8, due to [KT23]), it suffices to show how to break one-way puzzles using a PP oracle. Recall that a one-way puzzle is a pair of sampling and verification quantum algorithms $(\mathsf{Samp}, \mathsf{Ver})$ (see Definition 5.1.7). The algorithm $\mathsf{Samp}(1^n)$ outputs a pair (k, s), where k is referred to as the key and s as the puzzle. To break the one-way puzzle, it suffices to create a quantum algorithm $\mathcal{A}$ that, given a puzzle s, returns a key k' such that $\mathsf{Ver}(k', s)$ is accepted with non-negligible probability.

Our strategy to construct $\mathcal{A}$ is as follows. Given a puzzle s sampled according to Samp, we will sample a key k according to the conditional distribution of keys that are sampled together with s. This will suffice to break the one-way puzzle, as Ver accepts pairs from Samp with $(1 - \mathsf{negl})$-probability. To sample from this distribution, we first show how to use the PP oracle to estimate the conditional probability, and finally how to sample according to a distribution that is close to the true conditional distribution.

Lemma 5.3.1. *Let S be a (uniform) quantum polynomial time algorithm that outputs n bits, denoted as a pair of a string x, and a bit b. There exists a poly-time quantum algorithm $\mathcal{A}$ and a PP language $\mathcal{L}$ such that $\mathcal{A}^{\mathcal{L}}$ can estimate the distribution of bit b, conditioned on the output string x. Formally,*

$$p_{x,b} - \frac{1}{n^2} \leqslant \mathcal{A}^{\mathcal{L}}(S, 1^n, x, b) \leqslant p_{x,b} + \frac{1}{n^2}$$

where $p_{x,b} = \mathbb{P}\left[S(1^n) = (x, b) | S(1^n) \in \{(x, 0), (x, 1)\}\right]$.

Proof. Our quantum algorithm $\mathcal{A}$ will take as input the algorithm S, a unary description of the length 1^n, the outputs (x, b), and will estimate the conditional probability of b, conditioned on the first output of S being x.

Definition of the PP language. We will describe the PP language $\mathcal{L}$ in terms of a $\mathsf{PromisePostBQP}$ algorithm $\mathcal{B}(S, x, t)$. This will define a promise problem which is computed by $\mathcal{B}(S, x, t)$. By the equivalence $\mathsf{PromisePostBQP} = \mathsf{PromisePP}$, this gives us a promise problem in $\mathsf{PromisePP}$. Since $\mathsf{PromisePP}$ is a syntactic class, this can be extended to a language $\mathcal{L} \in \mathsf{PP}$. However, later on our algorithm $\mathcal{A}$ will only depend on the behaviour of the oracle to $\mathcal{L}$ on inputs that satisfy the promise condition of the promise problem defined by $\mathcal{B}(S, x, t)$.

We first assume without loss of generality that any measurements in S are delayed until the very end of the algorithm. The algorithm $\mathcal{B}$ will simulate S and postselect on the output measurements matching x. Conditioned on postselection succeeding, the output register for b will contain the pure state $\sqrt{p_{x,0}}\,|0\rangle + \sqrt{p_{x,1}}\,|1\rangle$. Then $\mathcal{B}$ measures the register of b, and repeats this procedure $r = \Theta(n^4)$ times, outputting 1 if the measurements yields 1 at least a $\frac{t}{2n^2}$ fraction of the time. This ends the description of the $\mathsf{PromisePostBQP}$ algorithm $\mathcal{B}(S, x, t)$.

Which inputs (S, x, t) are accepted by $\mathcal{B}$? Let $\mathbf{b}_1, \ldots, \mathbf{b}_r$ denote the random variables that correspond to the value of the b register at each simulation of S by $\mathcal{B}$. We define $\mathbf{B} = \sum_i \mathbf{b}_i$ to be their sum. Note that $\mathbf{B}/r$ is the fraction of measurements that yield a 1 in the algorithm $\mathcal{B}$, and recall that we accept if this fraction is at least $t/(2n^2)$. By standard concentration inequalities (e.g., Chebyshev's of Chernoff's; see [AB09, Lemma A.12 or Theorem A.14]), when the number of repetitions r is at least $\Theta(n^4)$, then

$$\mathop{\mathbb{P}}_{\mathbf{b}_1,\ldots,\mathbf{b}_r}\left[\left|\frac{\mathbf{B}}{r} - p_{x,1}\right| > \frac{1}{4n^2}\right] < \frac{1}{3}.$$

This implies that, if $p_{x,1} \geqslant \frac{t}{2n^2} + \frac{1}{4n^2}$, then $\frac{\mathbf{B}}{r} > \frac{t}{2n^2}$ with probability 2/3. Moreover, if $p_{x,1} \leqslant \frac{t}{2n^2} - \frac{1}{4n^2}$, then $\frac{\mathbf{B}}{r} < \frac{t}{2n^2}$ with probability 2/3. We conclude

$$\begin{cases} (S,x,t) \in \mathcal{L}, & \text{if} \quad p_{x,1} \geqslant \frac{t}{2n^2} + \frac{1}{4n^2}, \\ (S,x,t) \notin \mathcal{L}, & \text{if} \quad p_{x,1} \leqslant \frac{t}{2n^2} - \frac{1}{4n^2}. \end{cases} \tag{5.2}$$

Note that this only gives us information about the output of $\mathcal{B}$ on inputs (S, x, t) such that $\left|p_{x,1} - \frac{t}{2n^2}\right| \geqslant \frac{1}{4n^2}$. Henceforth we will call this inequality the *promise condition*. We remark that, if (S, x, t) does not satisfy the promise condition, then $(S, x, t+1)$ satisfies it. Indeed, if $\left|p_{x,1} - \frac{t}{2n^2}\right| < \frac{1}{4n^2}$, we have

$$p_{x,1} \leqslant \frac{t}{2n^2} + \frac{1}{4n^2} = \frac{t+1}{2n^2} - \frac{1}{2n^2} + \frac{1}{4n^2} = \frac{t+1}{2n^2} - \frac{1}{4n^2},$$

and therefore $\left|p_{x,1} - \frac{t+1}{2n^2}\right| \geqslant \frac{1}{4n^2}$. In particular, we obtain that $(S, x, t+1) \notin \mathcal{L}$.

Description of $\mathcal{A}^{\mathcal{L}}$. Given access to this PP language, $\mathcal{A}$ will query the oracle on the inputs $\{(S, x, 0), \ldots, (S, x, 2n^2)\}$. Then $\mathcal{A}$ will output $\frac{t}{2n^2}$, for the smallest t such that (S, x, t) is rejected by the oracle (if (S, x, t) is accepted for all t, then $\mathcal{A}$ outputs 1). Note that, if $(S, x, t-1)$ is accepted by the oracle, and (S, x, t) is rejected, then one of the following three things must have happened:

1. $(S, x, t-1) \in \mathcal{L}$ and $(S, x, t) \notin \mathcal{L}$, or

2. $(S, x, t-1)$ does not satisfy the promise condition, and $(S, x, t) \notin \mathcal{L}$, or

3. $(S, x, t-1) \in \mathcal{L}$ and (S, x, t) does not satisfy the promise condition.

As we observed above, it never occurs that $(S, x, t-1)$ does not satisfy the promise condition, and $(S, x, t) \in \mathcal{L}$. By inspection, using (5.2), in all of those three cases we get that the value $\frac{t}{2n^2}$ is an additive approximation to $p_{x,1}$ with error at most $\frac{1}{n^2}$. Additionally, since $p_{x,0} = 1 - p_{x,1}$, the value $\frac{2n^2-t}{2n^2}$ is an additive approximation for $p_{x,0}$. □

When $\mathsf{Samp}(1^n) \to (k, s)$ is is a uniform quantum polynomial-time algorithm outputting a pair of classical strings (k, s), we will denote by $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$ the distribution of keys k output by $\mathsf{Samp}(1^n)$, conditioned on the puzzle being s'.

Lemma 5.3.2. *Let* Samp *be a (uniform) quantum polynomial time algorithm such that* $\mathsf{Samp}(1^n)$ *outputs a pair of classical strings* (k, s)*, denoted as the key and the puzzle respectively, where* $k \in \{0,1\}^n$*. There exists a poly-time quantum algorithm* $\mathcal{A}$ *and a* PP *language* $\mathcal{L}$ *such that* $\mathcal{A}^{\mathcal{L}}$ *takes as input a puzzle* s' *and outputs a key* k'*, and whose distribution has total variation distance at most* $1/n$ *from* $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$*. In other words, we have*

$$\left(k' \mid k' \leftarrow \mathcal{A}^{\mathcal{L}}(1^n, s')\right) \approx_{\frac{1}{n}} (\mathsf{Samp} \,|\, s')^{\mathsf{key}}.$$

Proof. On a high level, the algorithm $\mathcal{A}$ will output a key k' by sampling its bits one by one. At the i^{th} iteration, it will use the first $i-1$ bits of k' to estimate the distribution of the i^{th} bit using a PP oracle, as in Lemma 5.3.1. Then it will sample the i^{th} bit of k' according to this estimated distribution. The PP language $\mathcal{L}$ that we use is the same as the one shown to exist in that lemma.

In more detail, let us define the sequence $\{\mathcal{S}^{\leqslant i}\}_{i \in [n]}$ of algorithms based on Samp, where $\mathcal{S}^{\leqslant i}$ is the (uniform) quantum polynomial time algorithm that simulates Samp, but only outputs the puzzle s and the first i bits of the key k.

On input $(1^n, s')$, the algorithm $\mathcal{A}$ will proceed in n iterations. In the first iteration, it will use Lemma 5.3.1 on $\mathcal{S}^{\leqslant 1}$ with output s' and estimate (up to $\frac{1}{n^2}$ additive error) the probability that the first bit of the key is 1, conditioned on the puzzle s'. Call this estimate $\tilde{p}_1$. The algorithm will then sample the first bit of k' according to the Bernoulli distribution defined by $\tilde{p}_1$.

Now $\mathcal{A}$ proceeds by sampling the remaining bits. In the i^{th} iteration, it uses Lemma 5.3.1 with the sampler $\mathcal{S}^{\leqslant i}$ and outputs an estimate $\tilde{p}_i$ of the probability that the i^{th} bit of k' is 1, conditioned on the puzzle s' and the first $i-1$ bits of

k'. Then it samples the i^{th} bit according to the estimated distribution. Finally, $\mathcal{A}$ outputs k' after the end of the n^{th} iteration.

It remains to show that the output distribution of $\mathcal{A}$ is close to $(\mathsf{Samp} \mid s')^{\mathsf{key}}$, the output distribution of $\mathsf{Samp}(1^n)$ conditioned on the puzzle s'. We will show this via a hybrid argument.

Let $\mathcal{D}_0 := (\mathsf{Samp} \mid s')^{\mathsf{key}}$ be the true distribution of the key k conditioned on the puzzle s'. We define n hybrid distributions $\mathcal{D}_i$ for $i \in \{1, \ldots, n\}$ on keys. The hybrid $\mathcal{D}_i$ runs in n iterations and in each iteration samples the next bit of the key. In the first i iterations (that correspond to the first i bits), the distribution $\mathcal{D}_i$ uses the estimated probabilities $\widetilde{p_1}, \ldots, \widetilde{p_i}$. The final $n - i$ bits are sampled according to the true conditional probabilities (i.e., according to $(\mathsf{Samp} \mid s')^{\mathsf{key}}$, conditioned on the outcome of the previous bits). Note that $\mathcal{D}_n$ now corresponds to the output distribution of our algorithm $\mathcal{A}$. We show that every two consecutive distributions are at most $\frac{1}{n^2}$ far in total variation distance, and thus the total distance between the true distribution of the key and the output distribution of $\mathcal{A}$ is at most $\frac{1}{n}$ from the triangle inequality.

Claim 5.3.3. *For every $i \in \{0, \ldots, n-1\}$, we have*

$$\mathsf{d}_{\mathsf{TV}}(\mathcal{D}_i, \mathcal{D}_{i+1}) \leqslant \frac{1}{n^2}.$$

Proof. Before diving into the proof, we introduce some useful notation. We use $k_{[x,y]}$ to denote the length-$(y - x + 1)$ substring that includes bits $\{x, \ldots, y\}$ of k. Additionally, even though $\mathcal{D}_i$ is a distribution over n-bit strings, we will abuse notation and consider the probability assigned to substrings of the form $k_{[x,y]}$. In that case, we write

$$\mathcal{D}_i(k_{[x,y]}) := \mathop{\mathbb{P}}_{k' \leftarrow \mathcal{D}_i} \left[k'_{[x,y]} = k_{[x,y]} \right].$$

We will also consider the probability assigned to substrings $k[x, y]$, conditioned on a prefix $k_{[1,x]}$, in which case we write

$$\mathcal{D}_i \left(k_{[x,y]} \middle| k_{[1,x-1]} \right) := \mathop{\mathbb{P}}_{k' \leftarrow \mathcal{D}_i} \left[k'_{[x,y]} = k_{[x,y]} \middle| k'_{[1,x-1]} = k_{[1,x-1]} \right].$$

With this notation in place, a direct calculation suffices for the $i = 0$ case. In particular, observe that

$$\begin{aligned} \mathcal{D}_1(k) &= \mathcal{D}_1 \left(k_{[1,1]} \right) \cdot \mathcal{D}_1 \left(k_{[2,n]} \middle| k_{[1,1]} \right) \\ &= \mathcal{D}_1 \left(k_{[1,1]} \right) \cdot \mathcal{D}_0 \left(k_{[2,n]} \middle| k_{[1,1]} \right), \end{aligned}$$

and similarly $\mathcal{D}_0(k) = \mathcal{D}_0\left(k_{[1,1]}\right) \cdot \mathcal{D}_0\left(k_{[2,n]} | k_{[1,1]}\right)$. Note that, because of Lemma 5.3.1, we have $\left|\mathcal{D}_0\left(k_{[1,1]}\right) - \mathcal{D}_1\left(k_{[1,1]}\right)\right| \leqslant 1/n^2$. The total variation distance satisfies:

$$\begin{aligned}
\mathsf{d}_{\mathsf{TV}}(\mathcal{D}_0, \mathcal{D}_1) &= \frac{1}{2} \sum_{k \in \{0,1\}^n} |\mathcal{D}_0(k) - \mathcal{D}_1(k)| \\
&= \frac{1}{2} \sum_{k \in \{0,1\}^n} \left|\mathcal{D}_0\left(k_{[1,1]}\right) \cdot \mathcal{D}_0\left(k_{[2,n]} | k_{[1,1]}\right) - \mathcal{D}_1\left(k_{[1,1]}\right) \cdot \mathcal{D}_0\left(k_{[2,n]} | k_{[1,1]}\right)\right| \\
&\leqslant \frac{1}{2} \sum_{k \in \{0,1\}^n} \left|\mathcal{D}_0\left(k_{[1,1]}\right) - \mathcal{D}_1\left(k_{[1,1]}\right)\right| \cdot \left|\mathcal{D}_0\left(k_{[2,n]} | k_{[1,1]}\right)\right| \\
&\leqslant \frac{1}{2n^2} \sum_{k \in \{0,1\}^n} \left|\mathcal{D}_0\left(k_{[2,n]} | k_{[1,1]}\right)\right| \\
&\leqslant \frac{1}{n^2}.
\end{aligned}$$

Let us now consider the distributions $\mathcal{D}_i, \mathcal{D}_{i+1}$. They both sample bits 1 up to i using the estimated probabilities, and bits $i+2$ up to n with the true conditional probabilities. Write $\mathcal{D}_{i+1}$ as

$$\begin{aligned}
\mathcal{D}_{i+1}(k) &= \mathcal{D}_{i+1}\left(k_{[1,i]}\right) \cdot \mathcal{D}_{i+1}\left(k_{[i+1,i+1]} | k_{[1,i]}\right) \cdot \mathcal{D}_{i+1}\left(k_{[i+2,n]} | k_{[1,i+1]}\right) \\
&= \mathcal{D}_{i+1}\left(k_{[1,i]}\right) \cdot \mathcal{D}_{i+1}\left(k_{[i+1,i+1]} | k_{[1,i]}\right) \cdot \mathcal{D}_i\left(k_{[i+2,n]} | k_{[1,i+1]}\right)
\end{aligned}$$

and similarly $\mathcal{D}_i$ as

$$\begin{aligned}
\mathcal{D}_i(k) &= \mathcal{D}_i\left(k_{[1,i]}\right) \cdot \mathcal{D}_i\left(k_{[i+1,i+1]} | k_{[1,i]}\right) \cdot \mathcal{D}_i\left(k_{[i+2,n]} | k_{[1,i+1]}\right) \\
&= \mathcal{D}_{i+1}\left(k_{[1,i]}\right) \cdot \mathcal{D}_i\left(k_{[i+1,i+1]} | k_{[1,i]}\right) \cdot \mathcal{D}_i\left(k_{[i+2,n]} | k_{[1,i+1]}\right).
\end{aligned}$$

Note that, because of Lemma 5.3.1, we have

$$\left|\mathcal{D}_i\left(k_{[i+1,i+1]} | k_{[1,i]}\right) - \mathcal{D}_{i+1}\left(k_{[i+1,i+1]} | k_{[1,i]}\right)\right| < 1/n^2.$$

The total variation distance satisfies:

$$
\begin{aligned}
&\mathsf{d_{TV}}(\mathcal{D}_i, \mathcal{D}_{i+1}) \\
&= \frac{1}{2} \sum_{k \in \{0,1\}^n} \left| \mathcal{D}_i(k) - \mathcal{D}_{i+1}(k) \right| \\
&\leqslant \frac{1}{2} \cdot \left(\sum_{k \in \{0,1\}^n} \mathcal{D}_{i+1}\left(k_{[1,i]}\right) \cdot \left| \mathcal{D}_i\left(k_{[i+1,i+1]} \middle| k_{[1,i]}\right) - \mathcal{D}_{i+1}\left(k_{[i+1,i+1]} \middle| k_{[1,i]}\right) \right| \right. \\
&\qquad\qquad \left. \cdot \mathcal{D}_i\left(k_{[i+2,n]} \middle| k_{[1,i+1]}\right) \right) \\
&\leqslant \frac{1}{2n^2} \sum_{k \in \{0,1\}^n} \mathcal{D}_{i+1}\left(k_{[1,i]}\right) \cdot \mathcal{D}_i\left(k_{[i+2,n]} \middle| k_{[1,i+1]}\right) \\
&= \frac{1}{2n^2} \sum_{k_{[1,i]} \in \{0,1\}^i} \mathcal{D}_{i+1}\left(k_{[1,i]}\right) \sum_{k_{[i+1,i+1]} \in \{0,1\}} \sum_{k_{[i+2,n]} \in \{0,1\}^{n-i-1}} \mathcal{D}_i\left(k_{[i+2,n]} \middle| k_{[1,i+1]}\right) \\
&= \frac{1}{n^2} \sum_{k_{[1,i]} \in \{0,1\}^i} \mathcal{D}_{i+1}\left(k_{[1,i]}\right) \\
&= \frac{1}{n^2}.
\end{aligned}
$$

□

As observed above, by the triangle inequality we obtain from the Claim that $\mathcal{D}_n$, which is the output distribution of our algorithm $\mathcal{A}$, has total variation distance at most $1/n$ from $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$. □

Theorem 5.3.4. *For any one-way puzzle* $(\mathsf{Ver}, \mathsf{Samp})$*, there exists a* PP *language* $\mathcal{L}$*, and a poly-time quantum algorithm* $\mathcal{A}^{\mathcal{L}}$ *such that*

$$
\mathop{\mathbb{P}}_{(k,s) \leftarrow \mathsf{Samp}(1^n)} \left[\mathsf{Ver}\left(\mathcal{A}^{\mathcal{L}}(s), s\right) = \top \right] \geqslant \frac{1}{2}.
$$

Proof. From the correctness property of the OWPuzzle, it holds that

$$
\mathop{\mathbb{P}}_{(k,s) \leftarrow \mathsf{Samp}(1^n)} \left[\mathsf{Ver}(k, s) = \top \right] \geqslant 1 - \mathsf{negl}(n).
$$

Recall that $(\mathsf{Samp} \,|\, s')^{\mathsf{key}}$ is the distribution over keys output by Samp, conditioned on the puzzle being equal to s'. It is clear that sampling the puzzle first, and then the key does not change their joint distribution, and thus

$$
\mathop{\mathbb{P}}_{\substack{(k,s) \leftarrow \mathsf{Samp}(1^n) \\ k' \leftarrow (\mathsf{Samp} \,|\, s)^{\mathsf{key}}}} \left[\mathsf{Ver}(k', s) = \top \right] \geqslant 1 - \mathsf{negl}(n).
$$

Given a puzzle s, Lemma 5.3.2 implies that there exists a quantum polynomial-time

algorithm $\mathcal{A}$ and a PP language $\mathcal{L}$, such that $\mathcal{A}^{\mathcal{L}}(s')$ outputs a key k' according to a distribution $\tilde{D}_{s'}$ such that

$$\tilde{D}_{s'} \approx_{1/n} (\mathsf{Samp} \,|\, s')^{\mathsf{key}}. \tag{5.3}$$

We have

$$\begin{aligned}
&\mathop{\mathbb{P}}_{\substack{(k,s)\leftarrow\mathsf{Samp}(1^n)\\ k'\leftarrow\tilde{D}_s}} \left[\mathsf{Ver}(k',s)=\top\right] \\
&\qquad = \sum_{s'} \mathop{\mathbb{P}}_{\substack{(k,s)\leftarrow\mathsf{Samp}(1^n)\\ k'\leftarrow\tilde{D}_s}} \left[\mathsf{Ver}(k',s)=\top \,\middle|\, s=s'\right] \cdot \mathop{\mathbb{P}}_{(k,s)\leftarrow\mathsf{Samp}(1^n)}[s=s'].
\end{aligned}$$

Now note that, conditioned on $s = s'$, the event $\{\mathsf{Ver}(k', s) = \top\}$ depends only on $k' \leftarrow \widetilde{D}_{s'}$, and thus we may use (5.3) to get

$$\begin{aligned}
&\mathbb{P}_{\substack{(k,s)\leftarrow\mathsf{Samp}(1^n)\\ k'\leftarrow\tilde{D}_s}} \left[\mathsf{Ver}(k',s)=\top\right] \\
&\geqslant \sum_{s'} \left(\mathop{\mathbb{P}}_{\substack{(k,s)\leftarrow\mathsf{Samp}(1^n)\\ k'\leftarrow(\mathsf{Samp}\,|\,s)^{\mathsf{key}}}} \left[\mathsf{Ver}(k',s)=\top \,\middle|\, s=s'\right] - 1/n \right) \cdot \mathop{\mathbb{P}}_{(k,s)\leftarrow\mathsf{Samp}(1^n)}[s=s'] \\
&= \sum_{s'} \left(\mathop{\mathbb{P}}_{\substack{(k,s)\leftarrow\mathsf{Samp}(1^n)\\ k'\leftarrow(\mathsf{Samp}\,|\,s)^{\mathsf{key}}}} \left[\mathsf{Ver}(k',s)=\top \,\middle|\, s=s'\right] \mathop{\mathbb{P}}_{(k,s)\leftarrow\mathsf{Samp}(1^n)}[s=s'] \right) \\
&- (1/n) \cdot \mathop{\mathbb{P}}_{(k,s)\leftarrow\mathsf{Samp}(1^n)}[s=s'] \\
&= \mathop{\mathbb{P}}_{\substack{(k,s)\leftarrow\mathsf{Samp}(1^n)\\ k'\leftarrow(\mathsf{Samp}\,|\,s)^{\mathsf{key}}}} \left[\mathsf{Ver}(k',s)=\top\right] - 1/n \geqslant 1 - 1/n - \mathsf{negl}(n) > 1 - 2/n.
\end{aligned}$$

This completes our argument. □

Now the desired result follows by combining Theorem 5.3.4 with Theorem 5.1.8 (Theorem 4.2 of [KT23]).

Corollary 5.3.5. *For any OWSG G of n-qubit states with key space $\mathcal{K}$ and with security parameter λ, there exists a* PP *language $\mathcal{L}$, a $\mathsf{poly}(\lambda)$-time quantum algorithm $\mathcal{A}^{\mathcal{L}}$, and $t = \mathsf{poly}(\lambda)$ such that*

$$\mathop{\mathbb{E}}_{\substack{k\leftarrow\mathcal{K}\\ k'\leftarrow\mathcal{A}^{\mathcal{L}}(|\phi_k\rangle^{\otimes t})}} \left[|\langle\phi_k|\phi_{k'}\rangle|^2 \right] \geqslant \frac{1}{2}.$$

Proof. The proof of Theorem 4.2 of [KT23] shows that, for every one-way state

generator G, there exists a one-way puzzle $P = (\mathsf{Samp}, \mathsf{Ver})$ such that, if there exists a quantum polynomial-time algorithm $\mathcal{A}$ that breaks P, then there exists a quantum polynomial-time algorithm $\mathcal{A}^*$ that breaks G. The corollary then follows by plugging the algorithm of Theorem 5.3.4 for P – a polynomial-time quantum algorithm with a PP oracle that breaks P – into their reduction. □

Chapter 6

Conclusions and open problems

In this thesis, we investigated a host of different computational models from the perspective of complexity-theoretic concerns, which we now briefly review. As we do so, we will pose a few open questions that could build on our contributions, or that are at least related to questions we investigated.

6.1 Comparator circuits

Our first investigation in Chapter 3 initiated the study of meta-computational problems for comparator circuits, and gave the first average-case lower bounds for this model, as we applied the shrinkage method to comparator circuits for the first time. We envision a few possible strengthenings and future applications of our techniques.

Algorithms and lower bounds for larger comparator circuits. Our lower bounds and circuit analysis algorithms only work for comparator circuits of size up to $n^{1.5-o(1)}$. Can we improve this? Specifically, can we show a lower bound of $n^{1.51}$ for comparator circuits computing a function of n bits, and design algorithms for comparator circuits of the same size? In Chapter 3, we used the random restriction method to analyse comparator circuits by shrinking the number of wires and using a structural result of [GR20] that relates the number of gates to the number of wires. Can we analyse the effect of random restrictions on the gates *directly*, and show a shrinkage lemma for comparator circuits on the number of gates, with a shrinkage exponent $\Gamma > 1/2$? Such a lemma would imply a lower bound that is better than $n^{1.5}$, and would allow us to design algorithms for comparator circuits larger than $n^{1.5}$.

Hardness magnification near the state-of-the-art. Recent work on *hardness magnification* [OS18, OPS19, CJW19, CHO+20] has shown that marginally improving the state-of-art worst-case lower bounds in a variety of circuit models would imply major breakthroughs in complexity theory. Although we don't prove this here, it is possible to show hardness magnification results for comparator circuits of size $n^{2+o(1)}$ by a simple adaptation of their arguments. Unfortunately, this does not match the best lower bounds we have for comparator circuits, which are around $n^{1.5-o(1)}$ as we have seen. Can we show a hardness magnification phenomenom nearly matching the state-of-art lower bounds for comparator circuits?

Extensions and restrictions of comparator circuits. Recent work of Komarath, Sarma and Sunil [KSS18] has provided characterisations of various complexity classes, such as L, P and NP, by means of extensions or restrictions of comparator circuits. Can our results and techniques applied to comparator circuits be extended to those variations of comparator circuits? Can this extension shed any light into the classes characterised by [KSS18]?

6.2 Constant-depth circuits vs. monotone circuits

In Chapter 4, we studied constant-depth circuits and monotone circuits, possibly the two most widely investigated models in circuit complexity theory. Although our results provide new insights about the relation between them, there are exceptionally basic questions that remain open.

Better monotone simulations of constant-depth circuits. While [Qui53] showed that negations can be efficiently eliminated from circuits of depth $d \leqslant 2$ that compute monotone functions, already at depth $d = 3$ the situation is much less clear. Theorem 4.3.3 (see Section 4.3.1) implies that every monotone function in depth-3 AC^0 admits a monotone circuit of size $2^{n-\Omega(n/\log^2 n)}$. It is unclear to us if this is optimal. While [COS17] rules out an efficient *constant-depth* monotone simulation, it is still possible (and consistent with Theorem 1.3.1) that $\mathsf{AC}^0_3 \cap \mathsf{Mono} \subseteq \mathsf{mNC}^1$. Is there a significantly better monotone circuit size upper bound for monotone functions computed by polynomial-size depth-3 circuits?

Constant-depth circuits vs. monotone circuits – a definite answer. Our results come close to solving the question posed by Grigni and Sipser [GS92]. Using our approach, it would be sufficient to show that $\mathsf{OddFactor}_n$ requires monotone

circuits of size $\exp(n^{\Omega(1)})$. This is closely related to the challenge of obtaining an exponential monotone circuit size lower bound for $\mathsf{Matching}_n$, a longstanding open problem in monotone complexity (see [Juk12, Section 9.11]).[1] Indeed, it's possible to reduce OddFactor to Matching using monotone AC^0 circuits (see [AK11, Lemma 6.18]).

Incidentally, the algebraic complexity variant of the AC^0 vs. $\mathsf{mSIZE}[\mathsf{poly}]$ problem has been recently settled in a strong way through a new separation result obtained by Chattopadhyay, Datta, and Mukhopadhyay [CDM21]. Could some of their techniques be useful to attack the more elusive Boolean case?

Non-Boolean monotone CSP dichotomy. We were able to give a full picture of which Boolean CSP problems are easy, and which are hard, for monotone circuit size and depth. Our results built upon existing reductions in the non-monotone case between different CSP problems, by means of polymorphisms. Recently, a full proof of the non-Boolean CSP dichotomy was obtained [Bul17, Zhu17]. Can we also give a dichotomy theorem for monotone circuits computing *non-Boolean* CSPs?

A cleverer application of lifting. Related to the two questions above is the final discussion in Section 4.4.5, wherein we explain that lifting theorems may yet yield a superpolynomial monotone circuit lower bound for a function in AC^0 by means of one of the two following approaches:

1. *By considering non-Boolean CSPs.* It's possible that, among the non-Boolean CSPs that are hard for monotone circuits, there exists one that is yet easy for non-monotone algorithms. Lower bounds for non-Boolean CSPs can also be obtained via lifting, and this may be one approach to the question of Grigni and Sipser [GS92].

2. *By considering a different interpolation of the generated instances.* We also discussed that the instances of the $\mathsf{CSP\text{-}SAT}_S$ problem generated by lifting do not cover the entirety of the minterms and maxterms of that function. In particular, it's possible to construct other monotone functions for which this technique implies an exponential monotone circuit lower bound. Is there one choice of these functions which lies in a non-monotone complexity class weaker than $\oplus\mathsf{L}$? For instance, if we obtained such a function in NL, by the same arguments involving depth-reduction and padding (see Lemma 4.1.1), the problem of Grigni and Sipser [GS92] would be solved.

[1] Note that in OddFactor we are concerned with the existence of a spanning subgraph where the degree of every vertex is odd, while in Matching the degree should be exactly 1.

A general theory on the power of cancellations. Finally, it would be interesting to develop a more general theory able to explain when cancellations can speedup the computation of monotone Boolean functions. Our investigation of monotone simulations and separations for different classes of monotone functions (graph properties and constraint satisfaction problems) can be seen as a further step in this direction.

6.3 Quantum one-way state generators

We showed two new results about one-way state generators. We proved that their existence is implied by almost all computational PRSs, and that $O(n)$-copy OWSGs can be broken by QPT algorithms with access to a PP oracle. These results bring the cartography of OWSGs much more in line with that of PRSs. We outline a few questions that remain open about the place of OWSGs in the space of quantum cryptography.

Separating OWSG and quantum bit commitments. Two recent papers [KT23] and [LMW23] make clear that proving that quantum bit commitments exist relative to a random oracle and any classical oracle (with access to the random oracle) would suffice to show a black box separation between OWSGs and quantum bit commitments. It follows from our results that a black box separation between quantum bit commitments and OWSGs can be achieved by proving a weaker statement, namely that there exists an oracle $\mathcal{O}$ relative to which quantum bit commitments exist and $\mathsf{PP}^{\mathcal{O}} \subseteq \mathsf{BQP}^{\mathcal{O}}$.

Since it is now known from [KT23] that OWSGs imply quantum bit commitments, proving this conjecture and the accompanying separation would provide strong evidence that quantum bit commitments are (at least among the quantum cryptographic primitives so far proposed) the minimal assumption for quantum cryptography.

Do OWSGs imply PRSs? We have now shown that PRSs imply OWSGs in nearly the greatest possible generality, and that the existence of either have equivalent known complexity ramifications. This gives additional motivation to the question of whether these primitives are equivalent (as their classical equivalents are known to be). Proving either an equivalence or separation (even for a limited copy

setting) would be an exciting accomplishment that would greatly clarify the cartography of quantum cryptography.

Optimal reduction from PRSs to OWSGs. OWSGs are inherently computational objects, meaning they cannot be implied by statistical primitives. Currently we know that

- PRSs with $m \geqslant \log n$ output qubits require computational assumptions and imply OWSGs.
- There exists a $c \in (0, 1)$ s.t. statistical PRSs with $m \leqslant c \log n$ output qubits exist. So the existence of PRSs in this regime cannot imply OWSGs.

This leaves the question open of whether a PRS with $m \in (c \log n, \log n]$ output qubits imply the existence of OWSGs. A possible tightening would be to show that, for any $c < 1$, a statistical PRS with $m \leqslant c \log n$ exists, though this would show the surprising result that just over 1 bit of randomness per amplitude is enough to achieve statistical closeness to Haar-random states.

Where are now all those Masters and Doctors whom you knew so well in their lifetime in the full flower of their learning? Other men now sit in their seats, and they are hardly ever called to mind. In their lifetime they seemed of great account, but now no one speaks of them. Oh, how swiftly the glory of the world passes away!

Thomas Kempis, *The Imitation of Christ* (c. 1420 AD)

Bibliography

[Aar05] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461:3473 – 3482, 2005. 40, 98

[AB87] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of boolean functions. *Combinatorica*, 7(1):1–22, 1987. 6, 15, 20, 76

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. 96, 97, 111

[ABI+09] Eric Allender, Michael Bauland, Neil Immerman, Henning Schnoor, and Heribert Vollmer. The complexity of satisfiability problems: Refining Schaefer's theorem. *J. Comput. Syst. Sci.*, 75(4):245–254, 2009. 22, 37, 63, 77, 79, 80, 81, 82, 83, 86, 88, 90, 92, 93, 147

[ADH97] Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM J. Comput.*, 26(5):1524–1540, 1997. 5, 28

[AG87] Miklós Ajtai and Yuri Gurevich. Monotone versus positive. *J. Assoc. Comput. Mach.*, 34(4):1004–1015, 1987. 6, 15, 17, 18, 37, 63

[AGQY22] Prabhanjan Ananth, Aditya Gulati, Luowen Qian, and Henry Yuen. Pseudorandom (function-like) quantum state generators: New definitions and applications. In *Theory of Cryptography Conference*, pages 237–265. Springer, 2022. 24, 28, 98, 105

[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. 65

[AK11] Jin Akiyama and Mikio Kano. *Factors and Factorizations of Graphs*, volume 2031 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. 121

[AKR+01] Eric Allender, Michal Koucký, Detlef Ronneburger, Sambuddha Roy, and V. Vinay. Time-space tradeoffs in the counting hierarchy. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 295–302. IEEE Computer Society, 2001. 36, 65, 66

[AKS83] M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *Symposium on Theory of Computing* (STOC), pages 1–9, 1983. 10, 70

[And85] Alexander E Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Doklady Akademii Nauk SSSR*, 282:1033–1037, 1985. 15, 20

[BB14] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, dec 2014. 23, 26

[BCG89] Mihir Bellare, Lenore Cowen, and Shafi Goldwasser. On the structure of secret key exchange protocols. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 604–605. Springer, 1989. 23

[BCQ22] Zvika Brakerski, Ran Canetti, and Luowen Qian. On the computational hardness needed for quantum cryptography. *arXiv preprint arXiv:2209.04101*, 2022. 25, 27, 108

[BCRV03] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with boolean blocks, part i: Post's lattice with applications to complexity theory. *SIGACT News*, 2003. 31, 37, 77, 78, 79, 90

[BCRV04] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with boolean blocks, part ii: Constraint satisfaction problems. *ACM SIGACT-Newsletter*, 35, 2004. 31, 37, 77, 79, 80, 86, 90, 91, 92

[BDHM92] Gerhard Buntrock, Carsten Damm, Ulrich Hertrampf, and Christoph Meinel. Structure and importance of logspace-mod class. *Math. Syst. Theory*, 25(3):223–237, 1992. 65, 68

[Ber82] S. J. Berkowitz. On some relationships between monotone and non-monotone circuit complexity. Technical Report. University of Toronto, 1982. 16

[BG99] Amos Beimel and Anna Gál. On arithmetic branching programs. *J. Comput. Syst. Sci.*, 59(2):195–220, 1999. 5

[BGW99] László Babai, Anna Gál, and Avi Wigderson. Superpolynomial lower bounds for monotone span programs. *Combinatorica*, 19(3):301–319, 1999. 15, 39, 66, 68, 85, 143

[BHH16] Fernando GSL Brandao, Aram W Harrow, and Michał Horodecki. Local random quantum circuits are approximate polynomial-designs. *Communications in Mathematical Physics*, 346:397–434, 2016. 104

[BHST14] Eric Blais, Johan Håstad, Rocco A. Servedio, and Li-Yang Tan. On DNF approximators for monotone boolean functions. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 235–246, 2014. 15

[BHvMW21] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021. 5

[BL88] Josh Cohen Benaloh and Jerry Leichter. Generalized secret sharing and monotone functions. In *Advances in Cryptology* (CRYPTO), pages 27–35, 1988. 14

[BS20] Zvika Brakerski and Omri Shmueli. Scalable pseudorandom quantum states. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 417–440. Springer, 2020. 24, 29, 101

[BST13] Eric Blais, Dominik Scheder, and Li-Yang Tan. Ajtai-gurevich redux. Manuscript. 2013. 15, 17, 37

[BT96] Nader H. Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996. 14

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006. 4

[Bul17] Andrei A. Bulatov. A dichotomy theorem for nonuniform csps. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 319–330. IEEE Computer Society, 2017. 7, 22, 90, 121

[Bul18] Andrei A. Bulatov. Constraint satisfaction problems: complexity and algorithms. *ACM SIGLOG News*, 5(4):4–24, 2018. 22

[Cab11] Enrique M. Cabaña. F distribution. In Miodrag Lovric, editor, *International Encyclopedia of Statistical Science*, pages 499–501. Springer, 2011. 102

[CDL01] Andrew Chiu, George I. Davida, and Bruce E. Litow. Division in logspace-uniform NC^1. *RAIRO Theor. Informatics Appl.*, 35(3):259–275, 2001. 3

[CDM21] Arkadev Chattopadhyay, Rajit Datta, and Partha Mukhopadhyay. Lower bounds for monotone arithmetic circuits via communication complexity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 786–799. ACM, 2021. 15, 121

[CFL14] Stephen A. Cook, Yuval Filmus, and Dai Tri Man Le. The complexity of the comparator circuit value problem. *ACM Trans. Comput. Theory*, 6(4):15:1–15:44, 2014. 4, 10

[CHO+20] Lijie Chen, Shuichi Hirahara, Igor C. Oliveira, Ján Pich, Ninad Rajgopal, and Rahul Santhanam. Beyond Natural Proofs: Hardness Magnification and Locality. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*, volume 151, pages 70:1–70:48, 2020. 13, 20, 36, 120

[CJW19] Lijie Chen, Ce Jin, and R. Ryan Williams. Hardness magnification for all sparse NP languages. In *Symposium on Foundations of Computer Science* (FOCS), pages 1240–1255, 2019. 13, 120

[CJW20] Lijie Chen, Ce Jin, and R. Ryan Williams. Sharp threshold results for computational complexity. In *Symposium on Theory of Computing* (STOC), pages 1335–1348, 2020. 34

[CKK+15] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Comput. Complex.*, 24(2):333–392, 2015. 11, 31, 32, 33, 47

[CKLM20] Mahdi Cheraghchi, Valentine Kabanets, Zhenjian Lu, and Dimitrios Myrisiotis. Circuit lower bounds for MCSP from local pseudorandom generators. *ACM Trans. Comput. Theory*, 12(3):21:1–21:27, 2020. 14, 34, 58, 60

[CKR20] Bruno Pasqualotto Cavalar, Mrinal Kumar, and Benjamin Rossman. Monotone circuit lower bounds from robust sunflowers. In *LATIN 2020: Theoretical Informatics - 14th Latin American Symposium, São Paulo, Brazil, January 5-8, 2021, Proceedings*, volume 12118 of *Lecture Notes in Computer Science*, pages 311–322. Springer, 2020. 75

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity classifications of Boolean constraint satisfaction problems*, volume 7 of *SIAM monographs on discrete mathematics and applications*. SIAM, 2001. 80

[Coo85] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Inf. Control.*, 64(1-3):2–21, 1985. 3

[COS17] Xi Chen, Igor C. Oliveira, and Rocco A. Servedio. Addition is exponentially harder than counting for shallow monotone circuits. In *STOC'17—Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1232–1245. ACM, New York, 2017. 15, 17, 18, 37, 120

[CZ16] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In *Symposium on Theory of Computing* (STOC), pages 670–683, 2016. 14

[DCEL09] Christoph Dankert, Richard Cleve, Joseph Emerson, and Etera Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A*, 80(1):012304, 2009. 25

[dRGR22] Susanna F. de Rezende, Mika Göös, and Robert Robere. Guest column: Proofs, circuits, and communication. *SIGACT News*, 53(1):59–82, 2022. 7, 15, 20, 35

[For03] Lance Fortnow. One complexity theorist's view of quantum computing. *Theor. Comput. Sci.*, 292(3):597–610, 2003. 7

[FV98] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. 68, 85, 144

[GC01] Daniel Gottesman and Isaac Chuang. Quantum digital signatures, 2001. 25, 26, 27

[GGLR98] Oded Goldreich, Shafi Goldwasser, Eric Lehman, and Dana Ron. Testing monotonicity. In *Symposium on Foundations of Computer Science,* (FOCS), pages 426–435, 1998. 14

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288. Springer, 1984. 8

[GI12] Oded Goldreich and Rani Izsak. Monotone circuits: One-way functions versus pseudorandom generators. *Theory Comput.*, 8(1):231–238, 2012. 16

[GII+19] Alexander Golovnev, Rahul Ilango, Russell Impagliazzo, Valentine Kabanets, Antonina Kolokolova, and Avishay Tal. $AC^0[p]$ lower bounds against MCSP via the coin problem. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 66:1–66:15, 2019. 14

[GJW18] Mika Göös, Rahul Jain, and Thomas Watson. Extension complexity of independent set polytopes. *SIAM J. Comput.*, 47(1):241–269, 2018. 14

[GKRS19] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *10th Innovations*

in Theoretical Computer Science, volume 124 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 38, 19. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2019. 6, 7, 10, 15, 18, 20, 21, 35, 36, 37, 38, 63, 66, 68, 85, 89, 143

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984. 8

[GMOR15] Siyao Guo, Tal Malkin, Igor C. Oliveira, and Alon Rosen. The power of negations in cryptography. In *Theory of Cryptography Conference* (TCC), pages 36–65, 2015. 16

[GR20] Anna Gál and Robert Robere. Lower bounds for (non-monotone) comparator circuits. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 58:1–58:13, 2020. 4, 10, 11, 14, 31, 32, 43, 45, 46, 119

[GS92] Michelangelo Grigni and Michael Sipser. Monotone complexity. In *Proceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, page 57–75, USA, 1992. Cambridge University Press. 6, 16, 18, 20, 63, 120, 121

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Symposium on Theory of Computing* (STOC), pages 6–20, 1986. 6, 31, 74, 75

[HIL+23] Shuichi Hirahara, Rahul Ilango, Zhenjian Lu, Mikito Nanashima, and Igor C. Oliveira. A duality between one-way functions and average-case symmetry of information. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1039–1050. ACM, 2023. 8

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 8

[Hir22a] Shuichi Hirahara. Meta-computational average-case complexity: A new paradigm toward excluding heuristica. *Bull. EATCS*, 136, 2022. 4

[Hir22b] Shuichi Hirahara. NP-hardness of learning programs and partial MCSP. In *Symposium on Foundations of Computer Science* (FOCS), 2022. 14

[HLW06] Patrick Hayden, Debbie W Leung, and Andreas Winter. Aspects of generic entanglement. *Communications in mathematical physics*, 265:95–117, 2006. 104

[HMMH+23] Jonas Haferkamp, Felipe Montealegre-Mora, Markus Heinrich, Jens Eisert, David Gross, and Ingo Roth. Efficient unitary designs with a system-size independent number of non-clifford gates. *Communications in Mathematical Physics*, 397(3):995–1041, 2023. 25

[HMY23] Minki Hhan, Tomoyuki Morimae, and Takashi Yamakawa. A note on output length of one-way state generators. *CoRR*, abs/2312.16025, 2023. 29

[HO02] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2002. 1

[Hra71] V. M. Hrapčenko. A certain method of obtaining estimates from below of the complexity of π-schemes. *Mat. Zametki*, 10:83–92, 1971. 9

[HWWY94] Johan Håstad, Ingo Wegener, Norbert Wurm, and Sang-Zin Yi. Optimal depth, very small size circuits for symmetric functions in AC^0. *Inf. Comput.*, 108(2):200–211, 1994. 70

[IKL+19] Christian Ikenmeyer, Balagopal Komarath, Christoph Lenzen, Vladimir Lysikov, Andrey Mokhov, and Karteek Sreenivasaiah. On the complexity of hazard-free circuits. *J. ACM*, 66(4):25:1–25:20, 2019. 14

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147, 1995. 23

[IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Symposium on Discrete Algorithms* (SODA), pages 961–972, 2012. 11, 31

[IMZ19] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):11:1–11:16, 2019. 11, 31, 34, 58, 59, 60

[INN+22] Sandy Irani, Anand Natarajan, Chinmay Nirkhe, Sujit Rao, and Henry Yuen. Quantum search-to-decision reductions and the state synthesis problem. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. 42

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 44–61, New York, NY, USA, 1989. Association for Computing Machinery. 23

[IRS21] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Hardness on any samplable distribution suffices: New characterizations of one-way functions by meta-complexity. *Electron. Colloquium Comput. Complex.*, TR21-082, 2021. 8

[JCG97] Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, jul 1997. 79

[Jea98] Peter Jeavons. On the algebraic structure of combinatorial problems. *Theor. Comput. Sci.*, 200(1-2):185–204, 1998. 37, 77, 79, 80, 92

[JKB95] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous univariate distributions. Vol. 2.* Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons, Inc., New York, second edition, 1995. A Wiley-Interscience Publication. 102

[JLL76] Neil D. Jones, Y. Edmund Lien, and William T. Laaser. New problems complete for nondeterministic log space. *Math. Syst. Theory*, 10:1–17, 1976. 93

[JLS18] Zhengfeng Ji, Yi-Kai Liu, and Fang Song. Pseudorandom quantum states. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part III 38*, pages 126–152. Springer, 2018. 23, 24, 99

[Joh03] Jan Johannsen. The complexity of satisfiability problems with two occurrences. Unpublished Manuscript, 2003. 39, 68, 69

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*. Springer, 2012. 2, 17, 51, 69, 91, 121

[KKL+20] Valentine Kabanets, Sajin Koroth, Zhenjian Lu, Dimitrios Myrisiotis, and Igor Carboni Oliveira. Algorithms and lower bounds for De Morgan formulas of low-communication leaf gates. In *Conference on Computational Complexity* (CCC), pages 15:1–15:41, 2020. 11, 14

[KR13] Ilan Komargodski and Ran Raz. Average-case lower bounds for formula size. In *Symposium on Theory of Computing* (STOC), pages 171–180, 2013. 11, 31, 32, 33

[Kra97] Jan Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symbolic Logic*, 62(2):457–486, 1997. 14

[Kre21] William Kretschmer. Quantum pseudorandomness and classical complexity. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 23, 25, 28, 31, 40, 41, 97, 98, 103, 104, 105

[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. 11, 32, 33

[KS92] Bala Kalyanasundaram and Georg Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4):545–557, 1992. 68

[KSS18] Balagopal Komarath, Jayalal Sarma, and K. S. Sunil. Comparator circuits over finite bounded posets. *Inf. Comput.*, 261:160–174, 2018. 10, 120

[KT23] Dakshita Khurana and Kabir Tomer. Commitments from quantum one-wayness. *CoRR*, abs/2310.11526, 2023. 23, 25, 26, 27, 28, 31, 40, 100, 101, 109, 110, 116, 122

[Kup15] Greg Kuperberg. How hard is it to approximate the jones polynomial? *Theory of Computing*, 11(6):183–219, 2015. 98

[Kup21] Denis Kuperberg. Positive first-order logic on words. In *Symposium on Logic in Computer Science* (LICS), pages 1–13, 2021. 15, 17, 37

[Kup22] Denis Kuperberg. Positive first-order logic on words and graphs. *CoRR*, abs/2201.11619, 2022. 15, 17, 37

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Structure in Complexity Theory Conference* (CCC), pages 102–111, 1993. 36, 68

[LC97] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Physical Review Letters*, 78(17):3410–3413, apr 1997. 23, 25, 109

[LMW23] Alex Lombardi, Fermi Ma, and John Wright. A one-query lower bound for unitary synthesis and breaking quantum cryptography. Cryptology ePrint Archive, Paper 2023/1602, 2023. 25, 28, 122

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and kolmogorov complexity. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1243–1254. IEEE, 2020. 8

[LT09] Benoît Larose and Pascal Tesson. Universal algebra and hardness results for constraint satisfaction problems. *Theor. Comput. Sci.*, 410(18):1629–1647, 2009. 22

[LV08] Ming Li and Paul M. B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Texts in Computer Science. Springer, 2008. 44

[MS92] Ernst W. Mayr and Ashok Subramanian. The complexity of circuit value and network stability. *J. Comput. Syst. Sci.*, 44(2):302–323, 1992. 4, 9, 10

[Mul59] Mervin E. Muller. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM*, 2(4):19–20, apr 1959. 98

[Mul87] Ketan Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Comb.*, 7(1):101–104, 1987. 3

[MY22a] Tomoyuki Morimae and Takashi Yamakawa. One-wayness in quantum cryptography. *arXiv preprint arXiv:2210.03394*, 2022. 24, 25, 27, 28, 29, 39, 99, 100, 103

[MY22b] Tomoyuki Morimae and Takashi Yamakawa. Quantum commitments and signatures without one-way functions. In *Annual International Cryptology Conference*, pages 269–295. Springer, 2022. 8, 23, 25, 40, 99, 101, 108

[NC16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016. 96, 97

[Nec66] È. I. Nechiporuk. On a Boolean function. *Dokl. Akad. Nauk SSSR*, 169:765–766, 1966. 10, 14

[Oko82] E. A. Okol'nishnikova. The effect of negations on the complexity of realization of monotone Boolean functions by formulas of bounded depth. *Metody Diskret. Analiz.*, (38):74–80, 1982. 6, 15, 17, 37, 63

[Oli13] Igor Carboni Oliveira. Algorithms versus circuit lower bounds. *CoRR*, abs/1309.0249, 2013. 11

[Oli15] Igor C. Oliveira. Unconditional lower bounds in complexity theory, 2015. (PhD Thesis, Columbia University). 16

[OPS19] Igor C. Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Computational Complexity Conference* (CCC), pages 27:1–27:29, 2019. 13, 120

[OS18] Igor C. Oliveira and Rahul Santhanam. Hardness magnification for natural problems. In *Symposium on Foundations of Computer Science* (FOCS), pages 65–76, 2018. 13, 120

[OSP23] Ryan O'Donnell, Rocco A. Servedio, and Pedro Paredes. Explicit orthogonal and unitary designs, 2023. 25, 28, 31, 40, 103, 109

[OSS19] Igor C. Oliveira, Rahul Santhanam, and Srikanth Srinivasan. Parity helps to compute majority. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, pages 23:1–23:17, 2019. 36, 66

[Pos41] Emil L. Post. *The Two-Valued Iterative Systems of Mathematical Logic. (AM-5)*. Princeton University Press, 1941. 37

[PR17] Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In *Symposium on Theory of Computing* (STOC), pages 1246–1255, 2017. 6, 75

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. Symb. Log.*, 62(3):981–998, 1997. 14

[Qui53] W. V. Quine. Two theorems about truth functions. *Bol. Soc. Mat. Mexicana*, 10:64–70, 1953. 16, 19, 73, 120

[Raz85a] A. A. Razborov. Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR*, 281(4):798–801, 1985. 14

[Raz85b] Alexander A Razborov. Lower bounds on monotone complexity of the logical permanent. *Mathematical Notes of the Academy of Sciences of the USSR*, 37(6):485–493, 1985. 15, 18, 20, 143

[Raz90] A. A. Razborov. Lower bounds on the complexity of realization of symmetric Boolean functions by gate switching circuits. *Mat. Zametki*, 48(6):79–90, 1990. 10

[Rei05] Omer Reingold. Undirected st-connectivity in log-space. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 376–385. ACM, 2005. 39, 65, 69

[Rez23] Susanna Rezende. Personal communication, 2023. 90

[RM99] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. 85, 90, 91

[Ros08a] Benjamin Rossman. Homomorphism preservation theorems. *J. ACM*, 55(3):Art. 15, 53, 2008. 16, 17, 20, 73, 76

[Ros08b] Benjamin Rossman. On the constant-depth complexity of k-clique. In *Symposium on Theory of Computing* (STOC), pages 721–730, 2008. 15

[Ros17a] Benjamin Rossman. An entropy proof of the switching lemma and tight bounds on the decision-tree size of AC^0. 2017. 73

[Ros17b] Benjamin Rossman. An improved homomorphism preservation theorem from lower bounds in circuit complexity. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 27:1–27:17, 2017. 16, 17, 20

[RPRC16] Robert Robere, Toniann Pitassi, Benjamin Rossman, and Stephen A. Cook. Exponential lower bounds for monotone span programs. In *Symposium on Foundations of Computer Science* (FOCS), pages 406–415, 2016. 4, 7, 10, 15

[RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006. 7

[RW92] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992. 15, 18, 70

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 216–226. ACM, 1978. 7, 22, 37, 63, 77, 79, 80, 90, 91, 92

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. 7

[SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discret. Math.*, 8(2):223–250, 1995. 59, 60

[ST17] Rocco A. Servedio and Li-Yang Tan. What circuit classes can be learned with non-trivial savings? In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 30:1–30:21, 2017. 11, 14, 61

[Sto95] Alexei P. Stolboushkin. Finitely monotone properties. In *Symposium on Logic in Computer Science* (LICS), pages 324–330, 1995. 15, 17, 37

[Sub61] B. A. Subbotovskaja. Realization of linear functions by formulas using $\vee$, &, $-$. *Soviet Math. Dokl., Soviet Mathematics. Doklady, 2*, pages 110–112, 1961. 31

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.*, 5(3-4):207–388, 2010. 4

[Tal15] Avishay Tal. #SAT algorithms from shrinkage. *Electron. Colloquium Comput. Complex.*, page 114, 2015. 11

[Tar88] É. Tardos. The gap between monotone and nonmonotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988. 15, 18, 20, 90

[Urq87] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. 69

[Val80] Leslie G. Valiant. Negation can be exponentially powerful. *Theor. Comput. Sci.*, 12:303–314, 1980. 16

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983. 4

[Wat18] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018. 98, 99, 103

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, jan 1983. 23, 26

[Wig94] Avi Wigderson. NL/poly $\subseteq \oplus$L/poly (preliminary version). In *Proceedings of the Ninth Annual Structure in Complexity Theory Conference, Amsterdam, The Netherlands, June 28 - July 1, 1994*, pages 59–62. IEEE Computer Society, 1994. 5

[Wil13] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM Journal on Computing*, 42(3):1218–1244, 2013. 12

[Wil14a] Ryan Williams. Algorithms for circuits and circuits for algorithms. In *Conference on Computational Complexity* (CCC), pages 248–261, 2014. 11, 12

[Wil14b] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM*, 61(1):2:1–2:32, 2014. 11, 12

[Yan22] Jun Yan. General properties of quantum bit commitments. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 628–657. Springer, 2022. 25, 27

[Zhu17] Dmitriy Zhuk. A proof of CSP dichotomy conjecture. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 331–342. IEEE Computer Society, 2017. 7, 22, 90, 121

Appendix A

Appendix to Chapter 3

A.1 An efficient simulation of B_2-formulas by comparator circuits

We observe in this section that general formulas can be simulated by comparator circuits with at most a constant factor blow-up on their size.

A *B_2-formula* is a Boolean formula whose gates are taken from the set of all 16 binary gates. The input leaves are labelled with literals (i.e., x_i or $\neg x_i$) or binary constants.

We first observe that, for every binary gate g, it's possible to construct a comparator with exactly 4 wires such that, when given $x, y, \neg x, \neg y$, it computes $g(x, y)$ and $\neg g(x, y)$.

Lemma A.1.1. *For every binary gate $g \in \{0,1\}^2 \to \{0,1\}$, there exists a comparator circuit $C_g : \{0,1\}^4 \to \{0,1\}^4$ such that one of the wires of $C_g(x, y, \neg x, \neg y)$ outputs $g(x, y)$ and another wire outputs $\neg g(x, y)$.*

Proof. We show something slightly stronger. We give four comparator circuits with 4 wires each, and such that all of the 16 Boolean functions $g : \{0,1\}^2 \to \{0,1\}$ are computed by one of the wires of these 4 circuits. Moreover, those circuits will have the property that, if $g(x, y)$ is computed by one of the wires, then so is $\neg g(x, y)$. One such circuit is the circuit computing $\oplus, \neg\oplus, 0, 1$ presented in the Introduction (Figure 1.3). Another simple such example is a circuit computing $x, y, \neg x, \neg y$, which does not have any gates. The remaining cases are shown in Figures A.1 and A.2 below.

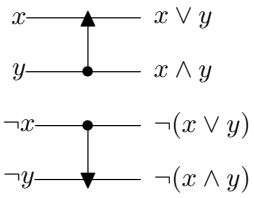


Figure A.1: A comparator circuit computing $\wedge, \vee, \neg\wedge, \neg\vee$.

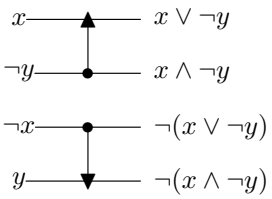


Figure A.2: A comparator circuit computing $x \vee \neg y, \neg(x \vee \neg y), x \wedge \neg y, \neg(x \wedge \neg y)$.

This completes the proof. □

We now show how the lemma allows us to deduce the reduction.

Theorem A.1.2. *For every B_2-formula with ℓ leaves and s gates, there exists a comparator circuit with 2ℓ wires and $O(s)$ gates computing the same function.*

Proof. We give a syntactic transformation from a B_2-formula F into an equivalent comparator circuit C_F with the desired efficiency.

Our transformation begins at the input layer. We create a new *negated* copy of each leaf of the formula F, giving us 2ℓ input wires in total in C_F. We then replace each binary gate g of F with a comparator circuit C_g computing $g(x, y)$ and $\neg g(x, y)$, proved to exist in Lemma A.1.1. We observe that this can be done by induction on the depth of the circuit. Every gate g at the bottom layer is connected with two input leaves x and y. Since we now have a negated copy of each leaf, we can input $x, y, \neg x, \neg y$ to C_g, and thus compute $g(x, y)$ and $\neg g(x, y)$. We observe that, since every input leaf is used only once in F, and we created a negated copy of every leaf, the same leaf will also not be used twice in C_F. On the induction step, a gate g will depend on two gates g_0 and g_1 at a lower layer, for which by induction we were able to compute $g_0, \neg g_0$ and $g_1, \neg g_1$. We input those values into C_g, thus computing g and $\neg g$.

Since each gate g is turned into a circuit C_g which has $O(1)$ comparator gates, we obtain $O(s)$ comparator gates at the end. □

Appendix B

Appendices to Chapter 4

B.1 A Lower Bound for 3-XOR-SAT Using the Approximation Method

As discussed in Section 1.3.3, [GKRS19] obtained an exponential lower bound on the monotone circuit size of the function 3-XOR-SAT using techniques from communication complexity and lifting. Here we observe that a weaker but still superpolynomial lower bound can be proved using the approximation method.

First, we recall the function $\mathsf{OddFactor}_n : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ of Section 4.2.2, which accepts a given graph if the graph contains an *odd factor*, which is a spanning subgraph in which the degree of every vertex is odd. For convenience, in this section we consider a weaker version of OddFactor, which takes as an input a *bipartite* graph with n vertices on each part, and accepts if the graph contains an odd factor. Let $\mathsf{Bipartite\text{-}OddFactor}_n : \{0,1\}^{n^2} \to \{0,1\}$ be this function. We remark that the lower bounds of Babai, Gál and Wigderson [BGW99] for OddFactor (Theorem 4.2.5) also hold for Bipartite-OddFactor. The proof of the monotone circuit lower bound in particular is essentially Razborov's lower bound for Matching via the approximation method [Raz85b].

Theorem B.1.1 ([BGW99])**.** *We have*

$$\mathsf{mSIZE}(\mathsf{Bipartite\text{-}OddFactor}_n) = n^{\Omega(\log n)}$$

and

$$\mathsf{mDEPTH}(\mathsf{Bipartite\text{-}OddFactor}_n) = \Omega(n).$$

We can reduce Bipartite-OddFactor to 3-XOR-SAT by noting that computing $\mathsf{Bipartite\text{-}OddFactor}_n(M)$ on a given matrix $M \in \{0,1\}^{n^2}$ is computationally equiv-

alent to deciding the satisfiability of the following $\mathbb{F}_2$ linear system over variables $\{x_{ij}\}$:

- For all $i \in [n]$: $\bigoplus_{k=1}^{n} x_{ik} = 1$;
- For all $j \in [n]$: $\bigoplus_{k=1}^{n} x_{kj} = 1$;
- For all $i, j \in [n]$ such that $M_{ij} = 0$: $x_{ij} = 0$.

We can then use a circuit for 3-XOR-SAT to solve this system by using a standard trick of introducing new variables to reduce the number of variables that appear in each equation, as done in the textbook reduction from SAT to 3-SAT. As the corresponding reductions turn out to be monotone, this implies monotone circuit and formula lower bounds for 3-XOR-SAT. We note that a somewhat similar argument (in the non-monotone setting) appears in Feder and Vardi [FV98, Theorem 30] regarding constraint satisfaction problems with the ability to count.

In order to formalise this argument, we will need the following definition and results.

Definition B.1.2. *Let f be a Boolean function. We define* $\mathsf{dual}(f) : x \mapsto \neg f(\neg x)$ *as the* dual *of f.*

Lemma B.1.3. *Let f be a monotone Boolean function. We have* $\mathsf{mSIZE}(f) = \mathsf{mSIZE}(\mathsf{dual}(f))$ *and* $\mathsf{mDEPTH}(f) = \mathsf{mDEPTH}(\mathsf{dual}(f))$.

Proof. The idea is to push negations to the bottom and eliminate double negations at the input layer. In other words, applying De Morgan rules, we can convert any $\{\wedge, \vee\}$-circuit computing f into a circuit computing $\mathsf{dual}(f)$ by swapping $\wedge$-gates for $\vee$-gates, and vice-versa. Moreover, this transformation preserves the depth of the circuit. □

We are ready to describe a monotone reduction from $\mathsf{Bipartite\text{-}OddFactor}_n$ to 3-XOR-SAT, which implies the desired lower bounds.

Theorem B.1.4. *There exists $\varepsilon > 0$ such that*

$$\mathsf{mSIZE}(\text{3-XOR-SAT}) = n^{\Omega(\log n)} \quad \textit{and} \quad \mathsf{mDEPTH}(\text{3-XOR-SAT}) = \Omega(n^{\varepsilon}).$$

Proof. Recall that the value of the function $\mathsf{Bipartite\text{-}OddFactor}_n(M)$ on a given matrix $M \in \{0,1\}^{n^2}$ is equal to 1 if the following system is satisfiable:

- For all $i \in [n]$: $\bigoplus_{k=1}^{n} x_{ik} = 1$;

- For all $j \in [n]$: $\bigoplus_{k=1}^{n} x_{kj} = 1$;
- For all $i, j \in [n]$ such that $M_{ij} = 0$: $x_{ij} = 0$.

We introduce some extra variables to reduce the number of variables in each equation in the following way. For every $i \in [n]$, introduce variables $z_{i1}, \ldots, z_{i(n-1)}$ and the equations

$$\begin{aligned} z_{i1} &= x_{i1} \oplus x_{i2}, \\ z_{i2} &= z_{i1} \oplus x_{i3}, \\ &\ldots \\ z_{i,(n-1)} &= z_{i,(n-2)} \oplus x_{i,n}, \\ z_{i,(n-1)} &= 1. \end{aligned}$$

Now note that these equations imply $z_{i,(n-2)} = \bigoplus_{k=1}^{n} x_{ik} = 1$. For each "column" equation $\bigoplus_{k=1}^{n} x_{kj} = 1$, we also add variables $w_{j1}, \ldots, w_{j(n-1)}$ as above. In total, we add at most $2n^2$ variables and $2n^2$ equations. Therefore, there is a system of linear equations on $O(n^2)$ variables, where each constraint contains at most 3 variables, which is satisfiable if and only if $\mathsf{Bipartite\text{-}OddFactor}_n(M) = 1$. Moreover, it is easy to see that the characteristic vector α of the set of equations of this system can be computed from M by an *anti-monotone* projection, as we activate a constraint that depends on the input when $M_{ij} = 0$.

Now let $f = \mathsf{dual}(\mathsf{3\text{-}XOR\text{-}SAT})$ and $\beta = \neg\alpha$. Since, by definition, 3-XOR-SAT accepts *unsatisfiable* systems, we get $\mathsf{Bipartite\text{-}OddFactor}_n(M) = \neg\mathsf{3\text{-}XOR\text{-}SAT}(\alpha) = f(\beta)$ and that β is a *monotone* projection of M. Therefore, by Lemma B.1.3, we obtain

$$\mathsf{mSIZE}(\mathsf{Bipartite\text{-}OddFactor}_n) \leqslant \mathsf{mSIZE}(\mathsf{3\text{-}XOR\text{-}SAT})$$

and

$$\mathsf{mDEPTH}(\mathsf{Bipartite\text{-}OddFactor}_n) \leqslant \mathsf{mDEPTH}(\mathsf{3\text{-}XOR\text{-}SAT}). \qquad \square$$

B.2 Background on Post's Lattice and Clones

In this section, we include the definitions of the various clones that are used in Chapter 4, as well as a figure of Post's lattice, which can be helpful when checking the proofs of Section 4.4.

Let $\to\colon (x, y) \mapsto (\neg x \vee y)$. Let also $\leftrightarrow\colon (x, y) \mapsto \neg(x \oplus y)$ and $\mathrm{id} : x \mapsto x$. Let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function. We say that f is *linear* if there exists

$c \in \{0,1\}^k$ and $b \in \{0,1\}$ such that $f(x) = \langle c, x\rangle + b \pmod 2$. We say that f is self-dual if $f = \mathsf{dual}(f)$. Let $a \in \{0,1\}$. We say that f is *a-reproducing* if $f(a,\ldots,a) = a$. We say that a set $T \subseteq \{0,1\}^k$ is *a-separating* if there exists $i \in [k]$ such that $x_i = a$ for all $x \in T$. We say that f is *a-separating* if $f^{-1}(a)$ is a-separating. We say that f is *a-separating of degree k* if every $T \subseteq f^{-1}(a)$ such that $|T| = k$ is a-separating. The *basis* of a clone B is a set of Boolean functions F such that $B = [F]$.

Name	Definition	Base
BF	All Boolean functions	$\{\vee, \wedge, \neg\}$
R_0	$\{f \in \mathrm{BF} : f \text{ is 0-reproducing}\}$	$\{\wedge, \oplus\}$
R_1	$\{f \in \mathrm{BF} : f \text{ is 1-reproducing}\}$	$\{\vee, \leftrightarrow\}$
R_2	$\mathrm{R}_1 \cap \mathrm{R}_0$	$\{\vee, x \wedge (y \leftrightarrow z)\}$
M	$\{f \in \mathrm{BF} : f \text{ is monotonic}\}$	$\{\vee, \wedge, 0, 1\}$
M_1	$\mathrm{M} \cap \mathrm{R}_1$	$\{\vee, \wedge, 1\}$
M_0	$\mathrm{M} \cap \mathrm{R}_0$	$\{\vee, \wedge, 0\}$
M_2	$\mathrm{M} \cap \mathrm{R}_2$	$\{\vee, \wedge\}$
$\mathrm{S}_0^{\mathrm{n}}$	$\{f \in \mathrm{BF} : f \text{ is 0-separating of degree } n\}$	$\{\rightarrow, \mathsf{dual}(h_n)\}$
S_0	$\{f \in \mathrm{BF} : f \text{ is 0-separating}\}$	$\{\rightarrow\}$
$\mathrm{S}_1^{\mathrm{n}}$	$\{f \in \mathrm{BF} : f \text{ is 1-separating of degree } n\}$	$\{x \wedge \overline{y}, h_n\}$
S_1	$\{f \in \mathrm{BF} : f \text{ is 1-separating}\}$	$\{x \wedge \overline{y}\}$
$\mathrm{S}_{02}^{\mathrm{n}}$	$\mathrm{S}_0^{\mathrm{n}} \cap \mathrm{R}_2$	$\{x \vee (y \wedge \overline{z}), \mathsf{dual}(h_n)\}$
S_{02}	$\mathrm{S}_0 \cap \mathrm{R}_2$	$\{x \vee (y \wedge \overline{z})\}$
$\mathrm{S}_{01}^{\mathrm{n}}$	$\mathrm{S}_0^{\mathrm{n}} \cap \mathrm{M}$	$\{\mathsf{dual}(h_n), 1\}$
S_{01}	$\mathrm{S}_0 \cap \mathrm{M}$	$\{x \vee (y \wedge z), 1\}$
$\mathrm{S}_{00}^{\mathrm{n}}$	$\mathrm{S}_0^{\mathrm{n}} \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \vee (y \wedge z), \mathsf{dual}(h_n)\}$
S_{00}	$\mathrm{S}_0 \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \vee (y \wedge z)\}$
$\mathrm{S}_{12}^{\mathrm{n}}$	$\mathrm{S}_1^{\mathrm{n}} \cap \mathrm{R}_2$	$\{x \wedge (y \vee \overline{z}), h_n\}$
S_{12}	$\mathrm{S}_1 \cap \mathrm{R}_2$	$\{x \wedge (y \vee \overline{z})\}$
$\mathrm{S}_{11}^{\mathrm{n}}$	$\mathrm{S}_1^{\mathrm{n}} \cap \mathrm{M}$	$\{h_n, 0\}$
S_{11}	$\mathrm{S}_1 \cap \mathrm{M}$	$\{x \wedge (y \vee z), 0\}$
$\mathrm{S}_{10}^{\mathrm{n}}$	$\mathrm{S}_1^{\mathrm{n}} \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \wedge (y \vee z), h_n\}$
S_{10}	$\mathrm{S}_1 \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \wedge (y \vee z)\}$
D	$\{f \in \mathrm{BF} : f \text{ is self-dual}\}$	$\{(x \wedge \overline{y}) \vee (x \wedge \overline{z}) \vee (\overline{y} \wedge \overline{z})\}$
D_1	$\mathrm{D} \cap \mathrm{R}_2$	$\{(x \wedge y) \vee (x \wedge \overline{z}) \vee (y \wedge \overline{z})\}$
D_2	$\mathrm{D} \cap \mathrm{M}$	$\{(x \wedge y) \vee (y \wedge z) \vee (x \wedge z)\}$
L	$\{f \in \mathrm{BF} : f \text{ is linear}\}$	$\{\oplus, 1\}$
L_0	$\mathrm{L} \cap \mathrm{R}_0$	$\{\oplus\}$
L_1	$\mathrm{L} \cap \mathrm{R}_1$	$\{\leftrightarrow\}$
L_2	$\mathrm{L} \cap \mathrm{R}$	$\{x \oplus y \oplus z\}$
L_3	$\mathrm{L} \cap \mathrm{D}$	$\{x \oplus y \oplus z \oplus 1\}$
V	$\{f \in \mathrm{BF} : f \text{ is constant or an } n\text{-ary OR function}\}$	$\{\vee, 0, 1\}$
V_0	$[\{\vee\}] \cup [\{0\}]$	$\{\vee, 0\}$
V_1	$[\{\vee\}] \cup [\{1\}]$	$\{\vee, 1\}$
V_2	$[\{\vee\}]$	$\{\vee\}$
E	$\{f \in \mathrm{BF} : f \text{ is constant or an } n\text{-ary AND function}\}$	$\{\wedge, 0, 1\}$
E_0	$[\{\wedge\}] \cup [\{0\}]$	$\{\wedge, 0\}$
E_1	$[\{\wedge\}] \cup [\{1\}]$	$\{\wedge, 1\}$
E_2	$[\{\wedge\}]$	$\{\wedge\}$
N	$[\{\neg\}] \cup [\{0\}] \cup [\{1\}]$	$\{\neg, 1\}$
N_2	$[\{\neg\}]$	$\{\neg\}$
I	$[\{\mathrm{id}\}] \cup [\{0\}] \cup [\{1\}]$	$\{\mathrm{id}, 0, 1\}$
I_0	$[\{\mathrm{id}\}] \cup [\{0\}]$	$\{\mathrm{id}, 0\}$
I_1	$[\{\mathrm{id}\}] \cup [\{1\}]$	$\{\mathrm{id}, 1\}$
I_2	$[\{\mathrm{id}\}]$	$\{\mathrm{id}\}$

Table B.2: Table of all closed classes of Boolean functions, and their bases. Here, h_n denotes the function $h_n(x_1, \ldots, x_{n+1}) = \bigvee_{i=1}^{n+1} \bigwedge_{j=1, j\neq i}^{n+1} x_j$. See Definition B.1.2 for the definition of $\mathsf{dual}(\cdot)$. The same table appears in [ABI+09, Table 1].