The Complexity of Joint Computation

by

Andrew Donald Drucker

B.A., Swarthmore College (2006)
S.M., Massachusetts Institute of Technology (2010)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

The Complexity of Joint Computation

by

Andrew Donald Drucker

B.A., Swarthmore College (2006)

S.M., Massachusetts Institute of Technology (2010)

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Joint computation is the ubiquitous scenario in which a computer is presented with not one, but *many* computational tasks to perform. A fundamental question arises: when can we cleverly combine computations, to perform them with greater efficiency or reliability than by tackling them separately? This thesis investigates the power and, especially, the *limits* of efficient joint computation, in several computational models: query algorithms, circuits, and Turing machines. We significantly improve and extend past results on limits to efficient joint computation for multiple independent tasks; identify barriers to progress towards better circuit lower bounds for multiple-output operators; and begin an original line of inquiry into the complexity of joint computation. In more detail, we make contributions in the following areas:

Improved direct product theorems for randomized query complexity: The "direct product problem" seeks to understand how the difficulty of computing a function on each of k independent inputs scales with k. We prove the following direct product theorem (DPT) for query complexity: if every T-query algorithm has success probability at most $1-\varepsilon$ in computing the Boolean function f on input distribution μ, then for $\alpha \leq 1$, every $\alpha\varepsilon Tk$-query algorithm has success probability at most $(2^{\alpha\varepsilon}(1-\varepsilon))^k$ in computing the k-fold direct product $f^{\otimes k}$ correctly on k independent inputs from μ. In light of examples due to Shaltiel, this statement gives an essentially optimal tradeoff between the query bound and the error probability. Using this DPT, we show that for an absolute constant $\alpha > 0$, the worst-case success probability of any $\alpha R_2(f)k$-query randomized algorithm for $f^{\otimes k}$ falls exponentially with k. The best previous statement of this type, due to Klauck, Špalek, and de Wolf, required a query bound of $O(bs(f)k)$.

Our proof technique involves defining and analyzing a collection of martingales associated with an algorithm attempting to solve $f^{\otimes k}$. Our method is quite general

and yields a new XOR lemma and threshold DPT for the query model, as well as DPTs for the query complexity of learning tasks, search problems, and tasks involving interaction with dynamic entities. We also give a version of our DPT in which decision tree size is the resource of interest.

Joint complexity in the Decision Tree Model: We study the *diversity* of possible behaviors of the joint computational complexity of a collection $f_1, \ldots, f_k$ of Boolean functions over a shared input. We focus on the deterministic decision tree model, with depth as the complexity measure; in this model, we prove a result to the effect that the "obvious" constraints on joint computational complexity are essentially the *only* ones.

The proof uses an intriguing new type of cryptographic data structure called a "mystery bin," which we construct using a polynomial separation between deterministic and unambiguous query complexity shown by Savický. We also pose a conjecture in the communication model which, if proved, would extend our result to that model.

Limitations of Lower-Bound Methods for the Wire Complexity of Boolean Operators: We study the circuit complexity of Boolean operators, i.e., collections of Boolean functions defined over a common input. Our focus is the well-studied model in which arbitrary Boolean functions are allowed as gates, and in which a circuit's complexity is measured by its depth and number of wires. We show sharp limitations of several existing lower-bound methods for this model.

First, we study an information-theoretic lower-bound method due to Cherukhin, which gave the first improvement over the lower bounds provided by the well-known superconcentrator technique for constant depths. (The lower bounds are still barely-superlinear, however) Cherukhin's method was formalized by Jukna as a general lower-bound criterion for Boolean operators, the "Strong Multiscale Entropy" (SME) property. It seemed plausible that this property could imply significantly better lower bounds by an improved analysis. However, we show that this is not the case, by exhibiting an explicit operator with the SME property that is computable in constant depths whose wire-complexity essentially matches the Cherukhin-Jukna lower bound (to within a constant multiplicative factor, for depths $d = 2, 3$ and for even depths $d \geq 6$).

Next, we show limitations of two simpler lower-bound criteria given by Jukna: the "entropy method" for general operators, and the "pairwise-distance method" for linear operators. We show that neither method gives super-linear lower bounds for depth 3. In the process, we obtain the first known polynomial separation between the depth-2 and depth-3 wire complexities for an explicit operator. We also continue the study (initiated by Jukna) of the complexity of "representing" a linear operator by bounded-depth circuits, a weaker notion than computing the operator.

New limits to classical and quantum instance compression: Given an instance of a decision problem that is too difficult to solve outright, we may aim for the more limited goal of compressing that instance into a smaller, equivalent instance of the same or a different problem. As a representative problem, say we are given Boolean formulas $\psi_1, \ldots, \psi_t$, each of length $n \ll t$, and we want to determine if at least one

ψ_j is satisfiable. Can we efficiently reduce this "OR-SAT" question to an equivalent problem instance (of SAT or another problem) of size poly(n), independent of t? We call any such reduction a "strong compression" reduction for OR-SAT. This would amount to a major gain from compressing $\psi_1, \ldots, \psi_t$ *jointly*, since we know of no way to reliably compress an individual SAT instance.

Harnik and Naor (FOCS '06/SICOMP '10) and Bodlaender, Downey, Fellows, and Hermelin (ICALP '08/JCSS '09) showed that the infeasibility of strong compression for OR-SAT would also imply limits to instance compression schemes for a large number of other, natural problems; this is significant because instance compression is a central technique in the design of so-called *fixed-parameter tractable* algorithms. Bodlaender et al. also showed that the infeasibility of strong compression for the analogous "AND-SAT" problem would establish limits to instance compression for another family of problems.

Fortnow and Santhanam (STOC '08) showed that deterministic (or 1-sided error randomized) strong compression for OR-SAT is not possible unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$; the case of AND-SAT remained mysterious. We give new and improved evidence against strong compression schemes for both OR-SAT and AND-SAT; our method applies to probabilistic compression schemes with 2-sided error. We also give versions of these results for an analogous task of *quantum instance compression*, in which a polynomial-time quantum reduction must output a quantum state that, in an appropriate sense, "preserves the answer" to the input instance. We give quantitatively similar evidence against strong compression for AND- and OR-SAT in this setting, albeit under less well-studied hypotheses about the relationship between NP and quantum complexity classes. To prove all of these results, we exploit the information bottleneck of an instance compression scheme, using a new method to "disguise" information being fed into a compressive mapping.

Thesis Supervisor: Scott Aaronson
Title: TIBCO Career Development Associate Professor
Department of Electrical Engineering and Computer Science

Acknowledgments

This thesis was a long time in the making, and accordingly is indebted to many people. First of all, I would like to thank my advisor, Scott Aaronson, for the innumerable forms of support he has shown me over the years it's been my privilege to know him. Scott has always treated me as a colleague and an equal. He has always enthusiastically invited me to explore the many wonderful ideas that have excited him over the years, and has taught me a huge amount about computational complexity and allied fields. At the same time he has placed faith in me to plot my own course of research.

My collaborations with Ronald de Wolf were in many ways akin to a second advisorship. In addition to sharing his scientific expertise, he generously shared his time to help me develop as a writer and editor. His formidable skills and work ethic have served as a model for me ever since.

I am grateful to Russell Impagliazzo for his kind support and guidance during my first year of graduate study at UC San Diego. I thank Michael Sipser and Scott Aaronson for each giving me valuable teaching opportunities at MIT, and for leading by example with their own inspiring teaching.

Fellow students have supported me and taught me a great deal over the years. Their guidance has come in many forms—from mathematical discussions, to practical advice about school and life. (I regard many of these talented individuals more as role models than as peers.) I have also valued their friendship and camaraderie through the academic process. A very partial list of fellow students who've given me valuable support and advice includes: Aleksander, Alexandr, Alex, Ankur, Ben, Bernhard, Cynthia, Jing, Krzysztof, Mayank, Michael, Nathan, Rotem, Swastik, Thomas.

It would be impossible to do justice here to the many friends outside of my academic sphere who've brought me happiness and emotional support during my graduate studies. I hope you know who you are and how much you've meant to me.

Most of all I am grateful to my whole loving family—especially my parents, Ron Drucker and Erica Buhrmann, for their unconditional love and support—and to Anna Torres, who was a loving partner, a steadfast friend, and a source of joy throughout my graduate studies.

My grandfather, Donald Buhrmann (1920-2012), was a gifted artist and a thoroughly remarkable man; he has always served as a model of achievement, purpose, and satisfaction in life for me. I am deeply saddened that he passed away shortly before this thesis was completed. I hope it would make him proud.

This thesis is based on the following papers:

- A. Drucker. **Multitask Efficiencies in the Decision Tree Model.** In: *IEEE Conference on Computational Complexity (CCC 2009).*

- A. Drucker. **Improved Direct Product Theorems for Randomized Query Complexity.** *Computational Complexity* 21(2), 2012 (special issue for CCC'11). Previous version in CCC'11; won Ronald V. Book award for Best Student Paper.

- A. Drucker. **Limitations of Lower-Bound Methods for the Wire Complexity of Boolean Operators.** In: *IEEE Conference on Computational Complexity (CCC)*, 2012. Won Ronald V. Book award for Best Student Paper.

- A. Drucker. **New Limits to Classical and Quantum Instance Compression.** To appear in: *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2012.

Contents

Chapter 1

Introduction

1.1 Efficient joint computation

Joint computation is the ubiquitous scenario in which a computer is presented with not one, but *many* computational tasks to perform. The question arises: when can we cleverly combine computations, to perform them with greater efficiency or reliability than by tackling them separately?

Important real-world examples of joint computation are easy to identify. The output of a typical piece of consumer software—say, a computer game—consists of a constantly-updating monitor display of hundreds of thousands of pixel values. The desired values tend to be highly correlated within spatial and temporal regions, leading to the strong potential for *"synergies,"* or joint savings, in their computation.

On a much larger scale, we also see striking examples of joint computation. Google, the world's largest online search engine, now processes billions of search queries a day worldwide. At any given time, a significant fraction of these queries are concentrated on a relatively small number of "hot" search terms. Such terms are identified and "preprocessed" to reduce the amount of computation per search.[1]

Turning to a more theoretical setting, we can regard a well-defined computational task as an example of joint computation whenever the desired output consists of

[1] For recent search-volume statistics, see [com10]. For more on how Google preprocesses the web, see, e.g., [Aus06].

more than one bit. In every such case, we have the *option* to compute these bits separately, but may benefit from computing them jointly. One caveat is that in numerical or algebraic problems, and in associated algebraic models of computation, it is often appropriate to regard a number (or field element) as "atomic." Thus over a ring R, we will consider a polynomial mapping like $t \to t^2$ as "single-output," whereas a mapping like $t \to (t, t^2, t^3)$ is distinctly "multiple-output." We will also use the term "operator" to refer to a mapping $F : S^n \to (S')^m$, where S, S' are finite or infinite sets and $m \geq 1$ (typically $S = S'$).

From this expansive viewpoint, joint computation is a pervasive theme in computing. Many natural computational problems of interest are multiple-output. Moreover, gains from joint computation have been at the heart of some of the most important and celebrated algorithms:

- *Sorting* is an operator mapping n integers $a_1, \ldots, a_n$ to n outputs—the same values, in sorted order. Sorting algorithms have been studied and implemented since before the dawn of the modern computing era [Knu73]. Fast sorting routines, such as MergeSort, Quicksort, and their relatives [Knu73], use $O(n \log n)$ integer comparisons, compared to the $\Theta(n^2)$ used by naïve approaches, and have led to enormous practical savings.

- The *discrete Fourier transform* (DFT) is a linear operator $F : \mathbb{C}^n \to \mathbb{C}^n$ that has numerous applications in science and engineering. While the obvious algorithm requires $\Theta(n^2)$ arithmetic operations over $\mathbb{C}$, the family of *Fast Fourier Transform* algorithms implements the same operator in $O(n \log n)$ arithmetic steps [FP11].

- *Matrix multiplication*, definable over any field or ring, is another fundamental operator with diverse applications. While naïve multiplication of two n-by-n matrices takes $\Theta(n^3)$ arithmetic steps, a series of ingenious algorithms beginning with Strassen [Str69] gave polynomial speedups for matrices with field elements. The current champion for asymptotic complexity, due to Vassilevska Williams [Wil12], uses $O(n^{2.373})$ arithmetic steps over any field. While this particular algorithm is not practically useful, techniques from this long line of work have led to dramatic speedups in practice as well.

An important observation is that, for each of the operators listed above, the complexity of the functions determining *individual* output values are well-understood, at least in the most natural computational model for the problem at hand:

- For sorting, it is known that for any $k \in [n]$, we can find the k^{th}-largest value among $a_1, \ldots, a_n$—for example, the median value—using $O(n)$ comparisons [BFP+73]. This is optimal up to a constant factor.

- For the DFT, any (fanin-two) algebraic circuit to compute a linear map $f : \mathbb{C}^n \to \mathbb{C}$ that depends nontrivially on all inputs must use $n - 1$ gates, simply to "gather" all the needed information in one place. Moreover, if scalar multiplications are free then $n - 1$ gates are sufficient.

- Similarly, to compute an individual matrix entry of the product of two n-by-n matrices requires an algebraic circuit of $2n - 1$ gates.

For each of these problems, the best known algorithm is faster—by polynomial factors—than the naïve approach of computing each output value separately. Thus, in a sense, the algorithmic improvements for these key problems have been "all about" efficient joint computation.

In spite of this, the concept of efficient joint computation is often omitted from discussions of important themes in the design of algorithms. The widely-used introductory algorithms textbook of Cormen *et al.* [CLRS09], which covers fast algorithms for sorting, median-finding, the DFT, and matrix multiplication, makes no explicit mention of joint computation as a unifying theme in these algorithms. Indeed, no standard, universally-recognized term seems to have emerged in the algorithms community for the concept of joint computation.

Why might this be? No conclusive answer can be given, but we may speculate. In algorithms research, computations involving more than one bit of output are the norm rather than the exception, and may be so familiar and common as to need no special designation. Techniques for multiple-output computation actually form an essential part of the toolkit for single-output problems as well, so that the study of these two classes of problems is closely integrated. For example, in the algorithm-design paradigm of *dynamic programming* [CLRS09, Chapter 15], to solve an instance of a computational problem we first "embed" it within a larger family of related problem instances, then solve *all of them* by an inductive approach that reuses information between the multiple computations.

While dynamic programming is widely-applicable, it does not encompass all efficient joint computation. It may be that joint computation is such a broad enterprise that powerful universal techniques simply do not exist. This would help explain the algorithms community's focus on developing effective tools for *particular* joint-computational problems.

1.2 Limits to computational synergies

Running alongside the important algorithmic developments described above, there has been a long tradition of significant research into the *complexity* of joint computation. That is, complexity theorists have tried to identify *inherent limits* to computational synergies for various multiple-output computational tasks. The present thesis falls within this tradition of study; we will review past work of this type in Sections 1.2 to 1.5, then describe our own contributions in Section 1.6.

At the outset, however, it seems fair to say that joint computation receives relatively little explicit, general discussion in complexity theory in comparison with other major themes. As in the algorithms community, no catch-all term for this concept is in wide use by researchers in complexity. When the complexity of joint computation is discussed, it is often in connection with two fairly specific questions—the so-called *direct sum* and *direct product* problems, which we will introduce in Section 1.3.

It seems likely that complexity theory's longstanding focus on *decision problems* as the usual objects of study has contributed to this state of affairs. Let us review the usual (folklore) justification for this focus. First, for most practically-interesting computational problems, the desired output is of length polynomially bounded in the input length. One can "reduce" the study of such functions to that of decision problems as follows. To any function $f : \{0,1\}^* \to \{0,1\}^*$ satisfying $|f(x)| \leq \text{poly}(|x|)$, we can associate a natural decision problem $L_f \subseteq \{0,1\}^*$. The input to L_f is a binary representation of a tuple $\langle x, i, b\rangle$, with $i \in \mathbb{N}, b \in \{0,1\}$; we define

$$L_f \ := \ \{\langle x, i, b\rangle : |f(x)| \geq i \text{ and } f(x)_i = b\} \ .$$

It is not hard to see that we can compute $f(x)$ using $\text{poly}(|x|)$ queries to an oracle for L_f, all on inputs $\langle x, i, b\rangle$ of length $|x| + O(\log |x|)$. Similarly, computing membership in L_f easily reduces to a single query to f itself. Thus, the complexity of computing f on input length n is "essentially the same" as that of computing L_f on input length $n + O(\log n)$, at least up to a polynomially-bounded multiplicative factor.

This does not give an exact equivalence between our function problem and the associated decision problem. However, for most problems of interest in complexity theory, such as NP-hard function and decision problems, there is currently a huge (superpolynomial) gap between the known upper and lower complexity bounds. From this perspective, the lack of exact equivalences between function and decision problems can be regarded as of secondary importance, and for this reason decision problems are often treated as acceptably general objects of study in complexity theory. (This focus has never been absolute, however; that would be a caricatured view of the field.)

The past several decades have seen an increasing theoretical interest in the "fine-grained" complexity of problems—particularly within computational models that allow extremely fast or economical computation, such as parallel algorithms [JáJ92]; "property testing" algorithms, which query only a small fraction of the input [Gol10]; and "streaming" algorithms, which use little storage space and make a small number of sequential passes over the input data [Mut05]. In these settings, where it is sometimes possible to prove asymptotically tight or nearly-tight bounds on the complexity of computational tasks, the known "equivalence" between a general function problem f and its "decision version" L_f (as described above) must be regarded as rather loose. Thus, our view is that taking a fine-grained approach to complexity theory, within any computational model, also motivates a detailed study of the complexity of multiple-output functions. We hope to see attention to this issue grow in years to come.

With that said, we emphasize that a great deal of interesting research has addressed the complexity of joint computation. We will now give a (selective) overview of past work in this area. To keep the discussion manageable and focused, we will only aim to describe our general state of knowledge for certain important computational models, including query algorithms, communication protocols, and various types of Boolean and algebraic circuits.[2] We will not attempt to treat every computational model, or to describe the known lower bounds for every specific multiple-output problem of practical interest.

In the complexity of joint computation, a useful if rough division can be made between two broad lines of research, according to whether the multiple computational tasks are defined with respect to *disjoint* inputs, or with respect to a *shared* input. The disjoint-inputs scenario is more specific, and the historical roots of its study are somewhat more recent. It has been influential within complexity theory, however, and is currently a very active area of study. We will review this area first.

1.3 Disjoint inputs

1.3.1 The disjoint-inputs intuition, and the direct sum and direct product problems

In the disjoint-inputs scenario, one studies operators of form

$$F(\overline{x}^1, \ldots, \overline{x}^k) \;=\; \big(f_1(\overline{x}^1), \ldots, f_k(\overline{x}^k)\big) \;:\; S^{k\times n} \to (S')^k \ ,$$

[2]The query model will be formally introduced in Chapter 2, and the circuit models relevant to our own results will be defined in Chapter 4. For background on communication models of computation, the reader may consult [KN96].

where no pair of input vectors $\overline{x}^i, \overline{x}^j$ share a variable in common. (We often consider families $\{F_n\}$ of such operators, one for each value of $n > 0$; in this case k may be a parameter depending on n.) Here, the natural algorithmic approach is to compute each $f_j(\overline{x}^j)$ separately, since the computational tasks appear to have nothing to do with each other. It is tempting to suspect that this approach is always optimal, i.e., that joint computation of $f_1, \ldots, f_k$ does not yield benefits. We will refer to this idea as the *"disjoint-inputs intuition."* As we will see, there are several ways to formalize it as a concrete hypothesis, even within a fixed computational model.

A great deal of research has explored the extent to which the disjoint-inputs intuition is actually valid. This work has found that in some settings the intuition can be confirmed completely; in other settings it can fail slightly, or fail badly; and in still other settings the extent of its validity remains unknown. Despite its fallibility, however, the disjoint-inputs intuition has been very fertile as a meta-hypothesis in complexity theory, and has helped to inspire an impressive range of research.

Before we review this work, it will be helpful to describe some fairly uninteresting senses in which the disjoint-inputs intuition fails; this will help clarify the proper focus of study. First, suppose that $\overline{x}^1, \ldots, \overline{x}^k$, while disjoint, are nevertheless *correlated* in some strong fashion. One way to model this is to assume that the inputs $\overline{x}^1, \ldots, \overline{x}^k$ do not take on arbitrary values, but are promised to obey some restriction; as a trivial example, they might always satisfy $\overline{x}^1 = \ldots = \overline{x}^k$. In this case, if we additionally have $f_1 = \ldots = f_m = f$, then we gain decisively from joint computation, since we need only evaluate f once and output m copies of the obtained value. This is clearly not the situation that our disjoint-inputs intuition aims to address. Thus in the disjoint-inputs scenario, if we do make a restriction on the admissible input-tuples $(\overline{x}^1, \ldots, \overline{x}^k)$ to our computational problem, we only consider cases where this restriction involves no dependence between the $\overline{x}^j$s, but can be expressed as a conjunction of restrictions, each involving a single $\overline{x}^j$. For simplicity's sake, however, in the review that follows we will assume that all functions are total, and no such restriction is made on the inputs.

As a similar failure of the disjoint-inputs intuition, if $\overline{x}^1, \ldots, \overline{x}^k$ are assumed to be generated by some *probability distribution* that involves significant dependence between the $\overline{x}^j$s, then we can enjoy the same kind of gains from joint computation. Thus, we will generally focus on distributional input-settings where the inputs are sampled from a *product distribution* with respect to the input blocks $\overline{x}^1, \ldots, \overline{x}^k$. That is, the distribution of each $\overline{x}^j$ is statistically independent of $\left(\overline{x}^{j'}\right)_{j' \neq j}$. We do allow, however, that the distribution over an *individual* input-vector $\overline{x}^j$ may be non-product.

For the purposes of high-level discussion, we will use "independent inputs" to refer, either to the setting where $(\overline{x}^1, \ldots, \overline{x}^k)$ are allowed to assume arbitrary values (with no distributional assumption made), or to the setting where these vectors are generated according to a product distribution with respect to the blocks $\overline{x}^1, \ldots, \overline{x}^k$. Following convention, we will refer to these input models as the *"worst-case"* and *"average-case"* (or *"distributional"*) input models, respectively.

There is another uninteresting way in which the disjoint-inputs intuition can fail: namely, it can fail if we measure an algorithm's cost in terms of its usage of a *reusable resource*, such as *space usage* by a Turing machine. For example, suppose we have functions $f_1(\overline{x}^1), f_2(\overline{x}^2) : \{0,1\}^n \rightarrow \{0,1\}$, each of which can be computed using n bits of space. Then we can also evaluate $f_1(\overline{x}^1), f_2(\overline{x}^2)$ using n bits of space, by just computing each output bit separately and clearing the storage tape before each computation. This effect, which occurs in both the disjoint-input and shared-input settings, can be viewed as a significant but conceptually-trivial gain from joint computation. Thus, our focus will be on *non-reusable* resources such as running time, or such as the number of circuit gates or wires used (in acyclic circuit models).

Once we focus attention to computational tasks where the $\overline{x}^j$s are "independent," in one of the two possible senses described above, the restriction that all f_js are equal to some single function f (applied to multiple, independent inputs) does not reduce the interest of the question, and does not seem to obscure any interesting issues. The known *counterexamples* to the disjoint-inputs intuition can all be realized within this restriction; also, results and conjectures become simpler to state and discuss when we make this restriction. As a result, many authors have done so. We will follow this practice, and will describe work in this area with attention to the case of a single function f, evaluated on k inputs; we note at the outset that some of the known results we describe either are proved for a more general setting involving multiple distinct functions f_j, or can be straightforwardly extended to such a setting.

If $f(\overline{x}) : S^n \to S'$ is any function and $k \in \mathbb{N}$, we let

$$f^{\otimes k} : \ S^{k \times n} \to (S')^k \ ,$$

the *k-fold tensor product* of f,[3] denote the mapping

$$f^{\otimes k}(\overline{x}^1, \ldots, \overline{x}^k) \ := \ \big(f(\overline{x}^1), \ldots, f(\overline{x}^k)\big) \ .$$

We can now describe the two major lines of research investigating the disjoint-inputs intuition. These two strands investigate two closely-related ways of formalizing the disjoint-inputs intuition; these two approaches are known as the *direct sum problem* and *direct product problem.*

As a rough initial description, in the *direct sum problem*, one tries to prove (or disprove) statements of the following form:

Suppose $f(\overline{x})$ requires cost T to compute "satisfactorily." Then, computing $f^{\otimes k}$ on k independent inputs requires cost T' to compute satisfactorily.

Such results, when true and provable, are known as *direct sum theorems.* Here, T' is determined by T, k, and possibly by other properties of the function f itself. The disjoint-inputs intuition suggests (sometimes falsely) that we may obtain a valid statement with $T' := Tk$.

To investigate the direct sum problem, one has to choose a notion of a "satisfactory" solution to a computational task. There are several options for what may count as satisfactory:

1. *Perfect solution:* an algorithm that computes $f(\overline{x})$ correctly on every admissible input $\overline{x}$ to f.

2. *Bounded-error solution for worst-case error:* a randomized algorithm that suc-

[3](also known as the *k-fold direct sum* or *direct product* of f)

ceeds "with high probability" on every admissible input $\overline{x}$.

3. *Distributional bounded-error solution:* a (deterministic or randomized) algorithm that succeeds with high probability when the input is drawn according to some particular input distribution $\mathcal{D}$. Here the success probability is taken both over $\mathcal{D}$, and over any randomness used by the algorithm itself. In this setting, we compare the complexity of computing f on inputs from $\mathcal{D}$ with the complexity of computing $f^{\otimes k}$ on $\mathcal{D}^{\otimes k}$, that is, on k inputs drawn independently from $\mathcal{D}$.

In the latter two cases, research on the direct sum question aims to compare the complexity of computing f with the complexity of computing $f^{\otimes k}$, where the success probability requirement defining a "satisfactory" solution is roughly *equivalent* for the two tasks. Sometimes the equivalence is not exact—results may compare the complexity of computing f with success probability .8 to the complexity of computing $f^{\otimes k}$ with success probability .9, say. (This sometimes makes results easier to prove; examples of this can be found, e.g., in [BBCR10, JKS10]). Results of this type, following items 2 or 3 above, can be stated in the following general form:

Suppose every algorithm using resources at most T has success probability at most p in computing f. Then, every algorithm using resources at most T' has success probability at most p' in computing $f^{\otimes k}$ on k independent inputs to f. (*)

Again, the probability may be with respect to a *worst-case* input model, or a *distributional* one. As we have noted, research on the direct sum problem focuses on the case $p \approx p'$, and aims to understand how large T' may be as determined by the other parameters and by f itself.

The direct sum problem can be contrasted with the *direct product problem.* The direct product problem also explores statements of form (*). A wide range of parameters is explored, but the characteristic focus is on proving results where p' decays exponentially as k grows; such results are conventionally referred to as *direct product theorems.* The division between direct sum and direct product problems is not exact, but it is fairly clear in practice and serves to indicate two distinct (but communicating) lines of research.

1.3.2 Background on the direct sum problem

Algebraic circuits

The direct sum problem was first studied in the context of algebraic circuits to compute *bilinear forms*, with principal attention to bilinear forms over fields. These are mappings

$$Q(\overline{x}, \overline{y}) : \ \mathbb{F}^{n_1+n_2} \to \mathbb{F}^m \ ,$$

where $\mathbb{F}$ is a field, and where Q is linear in each of $\overline{x}, \overline{y}$ whenever an input to the other argument is fixed. Such mappings are computable by circuits using addition and multiplication gates over $\mathbb{F}$; one natural complexity measure for a bilinear form, the *multiplicative complexity of* Q (which we'll denote by $C_{\text{mult}}(Q)$), is the least number of non-scalar *multiplication gates* used in any algebraic circuit to compute Q.

The *direct sum* $Q \oplus Q'$ of forms $Q(\overline{x}, \overline{y}), Q'(\overline{x}', \overline{y}')$ is the form that evaluates each of the pair on disjoint pairs of inputs. Strassen [Str73b] conjectured that the disjoint-inputs intuition always holds in perfect strength here: $C_{\text{mult}}(Q \oplus Q') = C_{\text{mult}}(Q) + C_{\text{mult}}(Q')$. This has been verified for many classes of forms [FZ77, AFW81, FW84, JT86, Bsh89] (see also [Bsh98]), but remains open in general. The conjecture *fails* for algebraic computation over general rings; this was proved by Schönhage [Sch81], and used as a tool in the development of improved algorithms for matrix multiplication. For a recent review of this line of work, see [Lan12].

Boolean circuits

The disjoint-inputs intuition can fail dramatically for Boolean circuits. The first example of this phenomenon was shown by Uhlig [Uhl74]. He showed that if $f : \{0,1\}^n \to \{0,1\}$ is *any* Boolean function, and if $r = r(n)$ satisfies $\log r = o\left(\frac{n}{\log n}\right)$, then we can compute $f^{\otimes r}$ using a Boolean circuit of $\frac{(1+o(1))2^n}{n}$ gates (over the basis $\{\wedge, \vee, \neg\}$). Remarkably, for randomly chosen f, this is asymptotically equal to the circuit size needed to compute a *single* copy of f! (See [Juk12, Chapter 1] for this latter fact.)

Another example of this type of behavior was shown, later but independently, by Paul [Pau76]. Both Uhlig and Paul's results only provide non-trivial information about functions with super-polynomial circuit complexity. However, also appearing in Paul's paper (and attributed to a referee) is an example that applies to polynomial-sized circuits. First, by a counting argument [Lup56], for each n there exists a linear transformation $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ that requires $\Omega\left(\frac{n^2}{\log n}\right)$ gates to compute in any (Boolean or $\mathbb{F}_2$-linear) circuit. Now consider $L^{\otimes n} : \mathbb{F}_2^{n\times n} \to \mathbb{F}_2^{n\times n}$; this operator corresponds to left-multiplication by L, and can be computed by an $\mathbb{F}_2$-linear circuit of size $n^{3-\Omega(1)} \ll n \cdot \frac{n^2}{\log n}$ using Strassen's algorithm [Str69] or any subsequent fast algorithm for matrix multiplication.

The disjoint-inputs intuition does at least hold for the important subclass of *monotone* Boolean circuits: Galbiati and Fischer [GG81] showed that the monotone circuit complexity $C_{\text{mon}}(f)$ satisfies $C_{\text{mon}}(f^{\otimes k}) = k \cdot C_{\text{mon}}(f)$.

Query algorithms and communication protocols

Let F be a (possibly non-Boolean) function defined over a Boolean input string $\overline{x}$, and let $D(F)$ denote the minimum number of queries to compute function F. It is simple to prove, and seems to be folklore, that the direct sum property holds in perfect strength for deterministic queries: for all F, k, we have $D(F^{\otimes k}) = k \cdot D(F)$. This result was generalized somewhat by Jain, Klauck, and Santha [JKS10]. The authors also prove a direct sum theorem for randomized query complexity with respect to the worst-case input model. Letting $R_\varepsilon(F)$ denote the cost of computing F with error probability at most ε on any input, they show that for any $0 < \varepsilon < \varepsilon'$, one has $R_{\varepsilon'}(F^{\otimes k}) = \Omega_{\varepsilon,\varepsilon'}\left(k \cdot R_\varepsilon(F)\right)$.

In the study of multiparty, distributed communication problems, an interesting example of an asymptotic savings from jointly computing functions of independent inputs was given by Stout [Sto86], in the model of "mesh computing with buses," a grid model of computation in which nodes have some limited broadcast capability. The example he exhibits is fairly natural.

In the standard model of two-party communication protocols, the direct sum problem was first raised by Karchmer, Raz, and Wigderson [KRW95], who showed that for the deterministic communication complexity $C(F)$ of a total function F, we have $C(F^{\otimes k}) = \Omega\left(k\sqrt{C(F)}\right)$ provided $C(F)$ is "reasonably large" with respect to the input size n. This result was independently obtained by Feder, Kushilevitz, Naor, and Nisan [FKNN95], who showed that for this result to hold, it suffices that $C(F) \geq 2\log n$. It remains open whether the $\sqrt{C(F)}$ factor in this result can be improved to $C(F)$.

Feder *et al.* prove this result by appealing to a known relation between the deterministic and nondeterministic communication complexity, and showing that the disjoint-inputs intuition holds in a strong way for the latter complexity measure. This technique, of analyzing the computational complexity of $F^{\otimes k}$ using a different, "surrogate" complexity measure of F, has been a frequent theme in the study of the direct sum and direct product problems. In the same vein, Karchmer et al. [KRW95] also gave a lower bound on $C(F^{\otimes k})$ in terms of the logarithm of the *rank* of the communication matrix for F.

Feder *et al.* also gave a strong direct product result for one-way deterministic communication complexity, as well as an example of a *failure* of the disjoint-inputs intuition: a *partial* function F where $C(F) = \Theta(\log n)$, yet $C(F^{\otimes k}) = O(k)$ for certain choices of $k \gg \log n$. For *randomized* protocols, they show that a similar failure of the disjoint-inputs intuition occurs for a total function, namely, the Equality function $EQ(\overline{x}, \overline{y}) := [\overline{x} = \overline{y}]$.

The next important development in the study of the direct sum problem for communication complexity came in the work of Chakrabarti, Shi, Wirth, and Yao [CSWY01], who gave a direct sum theorem for certain functions in the *simultaneous message* model of (randomized) communication protocols. Perhaps more important than the results were the techniques they developed, which identified a key complexity measure of a communication problem: its *informational complexity* (or *information cost*), which, roughly speaking, is defined as the minimal amount of information about the inputs that must be leaked to a third party eavesdropping on the communication channel. (Here, the minimum is taken over all protocols solving the communication problem with a desired success probability.) The authors showed that the information cost lower-bounds the communication complexity, and obeys a perfect direct sum theorem. Another key tool in this work was a method to *compress* communication protocols having small information cost.

The ideas in this work were extended in several subsequent papers; notably, [JRS03, HJMR10] obtained direct sum theorems for bounded-round randomized communication protocols. Recently Barak, Braverman, Chen, and Rao [BBCR10] made another significant advance using related ideas. Working with a slightly different notion of information cost—the *internal information cost*, in which we measure information being leaked *between* the two communicating parties rather than to an outside observer—and using different protocol-compression techniques, they showed new direct sum theorems that place no restriction on the number of rounds. These show that for $0 < \varepsilon < \varepsilon'$, the randomized communication complexity $R_\varepsilon(F^{\otimes k})$ in the distributional setting satisfies

$$R_\varepsilon(F^{\otimes k}) \cdot \log_2\left(R_\varepsilon(F^{\otimes k})\right) \;=\; \Omega_{\varepsilon,\varepsilon'}\left(\sqrt{k} \cdot R_{\varepsilon'}(F)\right) \ ,$$

for any input distribution over inputs $(\overline{x}, \overline{y})$ to F. For input distributions where $\overline{x}, \overline{y}$ are *independent*, they show that

$$R_\varepsilon(F^{\otimes k}) \cdot \text{polylog}\left(R_\varepsilon(F^{\otimes k})\right) \;=\; \Omega_{\varepsilon,\varepsilon'}\left(k \cdot R_{\varepsilon'}(F)\right) \ .$$

It is open whether this latter result holds when $(\overline{x}, \overline{y})$ are not independent. More recently, Braverman and Rao [BR11] have shown that this problem is essentially equivalent to a question about the communication complexity of the so-called *Correlated Pointer-Jumping* problem, a problem whose definition involves no immediate resemblance to the direct sum problem. Further connections between the computation of a function F on multiple instances and various measures of the information cost of F are made in [BR11, Bra12].

1.3.3 Background on the direct product problem

Recall that in the direct product problem, we study the validity of statements of the following form:

Suppose every algorithm using resources at most T has success probability at most p in computing f. Then, every algorithm using resources at most T' has success probability at most p' in computing $f^{\otimes k}$ on k independent inputs to f.

As mentioned earlier, in contrast to the direct sum problem for randomized algorithms, the direct product problem is distinguished by its focus on statements in which p' exhibits some form of exponential decay as k grows. The strength of a direct product theorem (or *DPT*) can be measured in terms of the dependence of the parameters T', p' on T, p, k, and, possibly, on the function f itself. From a lower-bounds perspective, we are interested in proving statements in which T' is as large and p' as small as possible, to establish that the k-fold problem is indeed "very hard."

There is also an important variant of the direct product problem, in which we are interested in computing the "k-fold XOR" $f^{\oplus k}(x^1, \ldots, x^k) := f(x^1) \oplus \ldots \oplus f(x^k)$ of k independent inputs to a Boolean function f; here $\oplus$ denotes the sum mod 2. An *XOR lemma* is a result which upper-bounds the success probability p' achievable by algorithms for $f^{\oplus k}$ using T' resources, under the assumption that any algorithm using T resources has success probability at most p.[4] An obvious difference from

[4]Terminology varies somewhat in the literature. For instance, what we call XOR lemmas are called "direct product theorems" in [Sha03], and what we refer to as direct product problems are in [Sha03] called the "concatenation variant."

DPTs is that in an XOR lemma, p' must always be at least $1/2$, since $f^{\oplus k}$ is Boolean and the algorithm could simply guess a random bit. The hope is that $(p' - 1/2)$ decays exponentially with k. Research on XOR lemmas has proceeded in parallel with research on direct product theorems; the known results are of similar strength (with some exceptions), and in some cases there are reductions known from XOR lemmas to DPTs or vice versa (see [Ung09, IK10] for an overview and recent results of this type).

The direct product problem has been studied extensively in models including Boolean circuits (e.g., [GNW95, IW97, IJKW10]), communication protocols [IRW94, Sha03, KŠdW07, LSŠ08, VW08], and query algorithms [IRW94, NRS99, Sha03, KŠdW07]. In all of these models, an optimal T-bounded algorithm which attempts to compute f can always be applied independently to each of k inputs, using at most $T' = Tk$ resources and succeeding with probability $p' = p^k$, so these are the "ideal," strongest parameters one might hope for in a DPT. However, direct product statements of such strength are generally false, as was shown by Shaltiel [Sha03], who gave a family of counterexamples which applies to all "reasonable" computational models. We will describe these examples (specialized to the query model) in Section 2.3 of Chapter 2.[5]

Thus, all DPTs shown have necessarily been weaker in one of several ways. First, researchers have restricted attention to algorithms of a special form. Shaltiel [Sha03] showed a DPT with the "ideal" parameters above holds for the query model, if the algorithm is required to query each of the k inputs exactly T times. He called such algorithms "fair."[6] A similar result for a special class of query algorithms called "decision forests" was shown earlier by Nisan, Rudich, and Saks [NRS99].

Second, DPTs have been shown for unrestricted algorithms, but using resource bounds whose strength depends on properties of the function f. These results require the resource bound T' to scale as $\mathcal{D}(f)k$, where $\mathcal{D}(f)$ is a complexity measure which

[5]Shaltiel calls a DPT "strong" if it applies to all p, T and its parameters satisfy $p' \leq p^{\Omega(k)}$ and $T' \geq \Omega(Tk)$. His counterexamples rule out strong DPTs for most computational models. In later works, the modifier "strong" has been used in a somewhat broader way. We will not use this terminology in this thesis.

[6]Actually, Shaltiel proved, in our terms, an optimal XOR lemma for fair algorithms, but as he noted, this implies an optimal DPT, and his proof method can also be modified to directly prove an optimal DPT for fair algorithms.

can be significantly smaller than the resources needed to compute a single instance of f. (We have already mentioned results of a similar type in the direct sum literature.) For example, Klauck, Špalek, and de Wolf [KŠdW07] (improving on earlier work of Aaronson [Aar05]) showed that for any f and any $\gamma > 0$, a DPT holds for f in which the achievable worst-case success probability p' is at most $(1/2 + \gamma)^k$, provided $T' \leq \alpha \cdot bs(f)k$ for some constant $\alpha = \alpha(\gamma) > 0$. Here $bs(f)$ is the *block sensitivity* of f [Nis91, BdW02], a complexity measure known to be related to the randomized query complexity by the inequalities $R_2(f)^{1/3} \leq bs(f) \leq R_2(f)$ (suppressing constant factors). Now, one can always compute f correctly on k instances with high probability using $O(R_2(f)k \log k)$ queries. For many functions, including random functions, $bs(f) = \Theta(R_2(f))$ so in these cases the DPT of [KŠdW07] gives a fairly tight result. However, examples are known [BdW02] where $bs(f) = O(\sqrt{R_2(f)})$, so the number of queries allowed by this DPT can be significantly less than one might hope.

Klauck, Špalek, and de Wolf also proved DPTs for *quantum* query algorithms computing f, in which the worst-case success probability p' drops exponentially in k if the number of allowed quantum queries is $O(\sqrt{bs(f)}k)$. For symmetric functions, direct product theorems of a strong form were proved for quantum query complexity by Ambainis, Špalek, and de Wolf [ASdW09]. Špalek [Š08] proved a DPT for quantum query algorithms where the resource bound T' scales in terms of a complexity measure called the *multiplicative quantum adversary*. Quite recently,[7] a sequence of works [She11, AMRR11, LR12] dramatically advanced our understanding of the direct product problem in the quantum query model. This culminated in a DPT for quantum queries due to Lee and Roland [LR12] in which the success probability decays exponentially even as the query bound scales as $\Omega(Q_2(f)k)$. Here, $Q_2(f)$ is the bounded-error quantum query complexity of a (possibly non-Boolean) function f.

In the model of communication protocols, several types of results have been shown. DPTs have been given for specific functions: e.g., in [KŠdW07] a DPT was proved for the quantum communication complexity of the Disjointness function, and a classical analogue was proved by Klauck [Kla10]. On the other hand, general DPTs have

[7](after a preprint of our paper [Dru12] appeared)

been given, whose resource bound scales in terms of complexity measures that may be significantly smaller than the communication complexity of f. For example, in communication complexity, DPTs have been shown whose strength is related to the so-called *discrepancy* of f [Sha03, LSŠ08].

Recently, there has been significant progress in the communication model. In the public-coin randomized setting, Jain showed a strong general-purpose DPT for one-way communication [Jai10a] and a DPT for two-way communication [Jai10b] whose strength depends on a new complexity measure (see also [JPY12]). Sherstov [She11] gave a new DPT for quantum communication, whose resource bound scales as $\Omega(\mathrm{GDM}_{1/5}(f)k)$, where $\mathrm{GDM}_{1/5}(f)$ is the lower bound on quantum communication complexity obtained by the *generalized discrepancy method*, the strongest lower bound technique known in the quantum setting.

In the Boolean circuit model, despite intensive study, the known results are quantitatively much weaker, and in particular require T' to *shrink* as k grows in order to make the success probability p' decay exponentially with k. One significant line of research in the circuit model has investigated direct product theorems in the circuit model, in which the assumption of full independence between the inputs to the various computations is replaced with weaker notions of independence. It was shown [Imp95, IW97] that, if these weakly-independent distributions are constructed appropriately, one can prove DPTs that approach the quantitative strength of the known results for the fully-independent case, while significantly reducing the *randomness complexity* needed to sample from these input distributions. This is a key ingredient in the "hardness versus randomness" approach to derandomizing probabilistic algorithms [BM84, Yao82, NW94, IW97], one of the most important developments in modern complexity theory (and also the most significant *application* of ideas from the direct product problem). This approach was initiated by Nisan and Wigderson [NW94] and carried further in many works; see [IW97] in particular, and [AB09] for an exposition for an overview of this area and further references.

1.4 Lower bounds for multiple functions of a shared input

The *shared-inputs scenario* is our chosen term for the general study of mappings

$$F(\overline{x}) : \; S^n \to (S')^m \; ,$$

typically with $S' = S$. This contains the disjoint-inputs scenario as a special case. Outside of that special case, research on the complexity of joint computation has aimed to exhibit explicit mappings which are as costly to compute as possible. Here, the main emphasis has been on *exact computation*—that is, on algorithms that compute the desired output with probability 1 on all inputs. Our review will focus on this setting. Another common focus in this research, which we will follow, is to consider cases where $m = \Theta(n)$.

1.4.1 The query and communication models

In the query model, it is straightforward to identify individual Boolean functions $f : \{0,1\}^n \to \{0,1\}$ with essentially maximal query complexity: for example, the PARITY function requires n queries for deterministic, nondeterministic, or randomized query algorithms. Considering multiple-output mappings $F : \{0,1\}^n \to \{0,1\}^{m>1}$ cannot increase the query complexity beyond n, since the trivial solution of reading the entire input always suffices to compute any mapping. Thus, the study of multiple-output mappings in the query model is uninteresting in this particular sense.

The situation for the communication model is very similar: we know how to prove nearly-maximal lower bounds $\Omega(n)$ on the communication needed to compute explicit Boolean functions $f(x, y)$ with $|x| = |y| = n$, and $2n$ bits of communication trivially suffices to compute any (Boolean or non-Boolean) mapping. In Chapter 3, we will explore a more promising line of research into the query complexity of joint computation in the shared-input setting, and also suggest a way to extend this research to the communication model.

1.4.2 Circuit models

For some circuit models of computation, any individual mapping $f : S^n \to S$ is "easy to compute," i.e., can be computed using $O(n)$ operations. As we have noted, this includes the comparison-based model as well as linear and bilinear arithmetic computations. It also includes a less well-known model, the *arbitrary-gates model*, which we will describe shortly. In these models, proving significant lower bounds *requires* attention to multiple-output operators.

Before reviewing the state of knowledge for these models, we will review part of the venerable history of lower-bounds research in the *monotone circuit* model. In this model, lower bounds for multiple-output monotone operators were studied intensively for a different reason: during a significant span of time (1971-1985), the multiple-output setting was the only setting in which super-linear lower bounds were known. More detailed surveys of the work on monotone circuits from this period can be found in the references [Kor03] and [Weg91], to which our brief review is indebted.

Monotone circuits

The first super-linear lower bounds for monotone circuits over the basis $\{\wedge, \vee\}$ were due to Nechiporuk [Nec71]. Using finite projective planes, Nechiporuk defined a system of $n = p^2$ disjunctions $f_1, \ldots, f_n$, each disjunction of size p over the Boolean input variables $x_1, \ldots, x_n$, and for which each pair $f_j, f_{j'}$ intersect in at most 1 variable. Nechiporuk proved that this property implies that optimal monotone circuits for computing $(f_1(\overline{x}), \ldots, f_n(\overline{x}))$ consist of $\vee$ gates only. This implies that joint computation of $f_1, \ldots, f_n$ yields no benefits, and gives a lower bound of $n(p-1) \sim n^{3/2}$ for the circuit size. Lower bounds of form $\Theta(n^{5/3})$ were proved by an elaboration of Nechiporuk's method in [Meh79, Pip80]. By somewhat similar methods, a lower bound of $\Theta(n^{3/2})$ for monotone circuits can be proved for other natural operators as well, such as the convolution and matrix multiplication operators over the semiring $\{\vee, \wedge\}$. In the case of matrix multiplication, the naïve circuit for this operator was shown to be exactly optimal (see [Weg91, Corollary 8.1]).

In a culmination of this line of work, Wegener (see [Weg91, Cor. 9.1]) exhibited a family of n simple DNFs $f_1, \ldots, f_n$ over $x_1, \ldots, x_n$, each individually computable with $O(n)$ gates, which require $\Omega\left(\frac{n^2}{\log n}\right)$ gates to compute jointly. Except for the $\frac{1}{\log n}$ factor, this is essentially the best lower bound one can attain from collections of n "simple" functions.

Prior to 1985, no super-linear lower bounds were known for the size of monotone circuits computing explicit monotone functions. That year, in a dramatic development, Razborov [Raz85a] proved *super-polynomial* lower bounds for the Clique decision problem; these lower bounds were subsequently improved to bounds of the form $2^{n^{\Omega(1)}}$ [And87, AB87]. While these results are now (justly) famous, few researchers today study the interesting previous work on monotone circuit lower bounds for multiple-output Boolean operators—a line of work that gave examples showing strong limits to joint computation in the monotone circuit model. Our view is that this earlier work's conceptual message deserves to be remembered.

Other "natural" restricted Boolean circuit models

Monotone circuits are the "natural," most-intuitive circuit model for computing monotone functions. Every monotone function has a monotone circuit over the basis $\{\wedge, \vee\}$, and in practice, ideas for computing monotone functions tend to be expressible as monotone circuits. It came as quite a surprise, then, when Razborov [Raz85b] showed that monotone circuits can be strongly sub-optimal for computing natural monotone functions such as the MATCH function, which detects whether a graph contains a perfect matching: this function has polynomial-size Boolean circuits, but not polynomial-size monotone circuits. (Less-dramatic polynomial separations between the monotone and non-monotone circuit complexity of monotone multi-output operators were shown earlier by Paul [Pau76]. In fact, such a separation is provided by the example of Boolean semiring matrix multiplication, which we have already encountered.)

We currently have no super-linear lower bounds for the non-monotone circuit complexity of explicit Boolean functions; this is one of the biggest embarrassments in the field. It is even open to prove a super-linear lower bound for the size of log-depth circuits to compute any explicit operator $f : \{0,1\}^n \to \{0,1\}^n$. Thus the state of our knowledge for non-monotone circuit complexity is poor even when compared with our understanding of the monotone circuit model in 1971 (after the pioneering work of Nechiporuk [Nec71]).

Thus, one thrust of research in circuit complexity has been to identify and analyze circuit models that represent "natural" algorithmic paradigms for particular classes of Boolean functions and operators, by analogy with the study of monotone circuits for monotone functions. Multiple-output operators have played a prominent role in this area.

One class of "natural" circuits for problems related to the Sorting problem are the so-called "conservative" circuits [PV76]. Loosely speaking, these circuits (a family of related models) treat certain input elements as "atomic" and do not modify them, but instead "route" them through the circuit over the course of a computation. Within this model, sorting n elements by comparisons is well-known to require $\Omega(n \log n)$ operations, even when the n input elements are all bits.

Another prominent trend across circuit complexity (also motivated by the difficulty of proving general lower bounds) has been to try to better understand the circuit complexity of *bounded-depth* circuits, allowing unbounded fanin of gates. This trend has appeared in the study of conservative circuits, and we will sketch one notable lower bound that has been obtained for a multiple-output operator; this will be useful as a benchmark of comparison when we describe the much weaker known results for more general classes of circuits.

The *Boolean shift operator* [PV76], denoted $\mathrm{shift}_n : \{0,1\}^{n+\lceil \log n \rceil} \to \{0,1\}^n$, is defined as follows. We are given a length-n string $\overline{x}$, indexed as $\overline{x} = (x_0, \ldots, x_{n-1})$. We regard a second input string $i \in \{0,1\}^{\lceil \log n \rceil}$ as a value in $\mathbb{Z}_n$, and define

$$\mathrm{shift}_n(\overline{x}, i) \ := \ (x_{-i}, x_{1-i}, \ldots, x_{n-1-i}) \ ,$$

with index arithmetic taken mod n. Thus the input $\overline{x}$ is cyclically shifted by an amount i. In the natural circuit-design paradigm for this problem, the circuit becomes a *routing device* for each fixed setting to i; the bits of $\overline{x}$ are not inspected, but merely routed to their "destinations" in the output. The best currently-known circuits for this problem follow this *routing paradigm* [PV76, PY82]. For circuits obeying the routing paradigm, and in which we allow the circuit to "preprocess" the input i for free, we understand the complexity of this operator rather precisely: in depth $d = d(n) \geq 2$, the circuit complexity is at least $d \cdot n^{1+1/d}$, and at most $\widetilde{O}(n^{1+1/d})$ [PY82]. For unbounded depth, the complexity is $\Theta(n \log n)$ [PV76].

Thus, the Boolean shift operator is an example of an operator for which we benefit substantially from joint computation of the multiple output bits—and all the more so as we increase the allowed depth—but for which we can also identify meaningful limits to joint computation in the natural associated computing paradigm. As we will see, we have been not been so fortunate in the study of more general circuits. We have also had very limited success in proving lower bounds for the size of linear circuits, the "natural" computational model for linear operators.

More constant-depth circuits: AC^0 and TC^0

As mentioned, no super-linear circuit lower bounds over a complete basis are known for explicit Boolean functions. We briefly review what is known about constant-depth, non-monotone circuits, where very little research seems to have probed the issue of joint computation. For constant-depth, unbounded-fanin circuits over the basis $\{\vee, \wedge, \neg\}$ (also known as AC^0 circuits), Håstad [Hås86], improving on earlier work [FSS84, Yao85], proved very strong super-polynomial lower bounds for the number of gates required to compute a simple explicit function—the PARITY function. Given this happy state of affairs, there seems to have been little research into the question of joint computation for AC^0 circuits; to the best of our knowledge there is no known example of an explicit operator $F : \{0,1\}^n \rightarrow \{0,1\}^n$ whose depth-d AC^0 circuit complexity is asymptotically greater than the largest depth-d complexity of any of its component functions (for large constants d). From our perspective, this omission suggests a direction for future work.

For the richer model of so-called TC^0 circuits, which are constant-depth circuits consisting of (weighted) *threshold gates*, our known lower bounds for Boolean functions are much weaker: the best result is due to Impagliazzo, Paturi, and Saks [IPS97] who show that the PARITY function requires at least $\Omega(n^{1+\alpha^d})$ wires to compute in depth d, for an absolute constant $0 < \alpha < 1$. A super-linear lower bound for *gates* is not known. In this model, it might be feasible to prove higher lower bounds for multiple-output operators, but again we are not aware of work in this direction.

Beyond TC^0 circuits, researchers have explored a physically unrealistic, but powerful and interesting, circuit model called the *arbitrary gates* model. This latter model is tailor-made to explore questions of efficient joint computation. We will review it next, along with the model of linear algebraic circuits over $\mathbb{F}_2$ (since the known lower bounds for these two models are similar). Our review of this area will be rather detailed, as this provides needed background for our original contributions in Chapter 4.

1.4.3 The arbitrary-gates and linear algebraic circuit models

A great deal of work, including the papers [Val76, Val77, DDPW83, CFL83, CFL85, Pud94, PR94, RS03, Che08a, Juk10a, Juk10b, JS10], has studied the circuit model in which unbounded fanin is allowed, and in which circuit gates can apply *arbitrary* Boolean functions to their inputs. In this model, we study the number of *wires* required in such a circuit to compute an operator F, a quantity we denote as $s(F)$. In our discussion we will focus attention on operators F of n input and $\Theta(n)$ output bits, since this is the focus of most prior work and seems to capture most interesting issues.

While allowing gates to compute arbitrary Boolean functions is not realistic, there are a number of motivations to study this model. First, it arguably provides a natural measure of the "information complexity" of Boolean operators. Second, lower bounds in this strong circuit model are highly desirable, since they also apply to a variety of more realistic models. Third, several natural circuit lower-bound criteria apply even to circuits with arbitrary gates, and it seems worthwhile to understand how far techniques of this kind can carry us. Finally, for at least one important class of Boolean operators—the $\mathbb{F}_2$-linear operators, naturally computable by $\mathbb{F}_2$-linear circuits—it remains unclear whether allowing arbitrary gates in our circuits even confers additional power.

Any individual Boolean function can by trivially computed with n wires in the arbitrary-gates model, so n^2 wires always suffice to compute an operator $F : \{0,1\}^n \to \{0,1\}^n$. In general, this is not far from optimal: random (non-linear) operators require $\Omega(n^2)$ wires to compute [JS10]. Thus random collections of Boolean functions are, in a sense, "computationally orthogonal" to one another. It would be extremely interesting to identify an *explicit* function collection with this property; however, proving a super-linear lower bound $s(F) = \omega(n)$ for an explicit operator F is a long-standing open problem.

This has led researchers to consider circuits with arbitrary gates but restricted *depth*. Even depth-2 circuits in this model are powerful, and their study was strongly motivated by work of Valiant [Val77] (see [Vio09]), who showed that any operator with depth-2 wire complexity $\omega(n^2/\ln\ln n)$ also cannot be computed by linear-size, logarithmic-depth Boolean circuits (of fanin 2). However, the known lower bounds for depth 2 are too weak to apply Valiant's results. For depth-2 circuits, the best bounds for explicit operators are of form $\Omega(n^{3/2})$ [Che08a, Juk10a]. For depths 3 and 4, the best bounds are $\Omega(n \ln n)$ and $\Omega(n \ln\ln n)$ respectively [Che08a]; for higher constant depths the known bounds (described in Section 4.1.1) are barely super-linear [DDPW83, Pud94, Che08a].

One might suspect that the difficulty of proving strong lower bounds stems from the unrealistic strength of the circuit model being studied. A seemingly much more modest aim is to prove lower bounds in the *linear algebraic circuit* model over $\mathbb{F}_2$. In this model, we require the circuit gates to compute $\mathbb{F}_2$-linear functions, i.e., sums mod 2; we again allow unbounded fanin. Given some linear operator $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we let $s^{\oplus}(L)$ denote the number of wires needed to compute L with a linear circuit. Lupanov [Lup56] (and later Bublitz [Bub86]) showed that $s^{\oplus}(L) = O(n^2/\ln n)$, and that this bound is tight if L is chosen randomly.

Unfortunately, the known lower bounds for *explicit* linear operators in the linear circuit model are just as discouragingly weak as for operators in the arbitrary-gates model. Moreover, since the lower bounds quoted earlier were shown for non-linear operators, the situation is actually slightly *worse* in the linear case: for example, for depth-2 circuits, the best known lower bound for an explicit linear operator is $\Omega\left(n\left(\frac{\ln n}{\ln\ln n}\right)^2\right)$, proved very recently [GHK$^+$12].

Thus, it is a major unmet challenge to develop lower-bound techniques that effectively exploit the specific behavior of linear circuits.[8] In fact, it is an open question whether $s^{\oplus}(L)$ can be noticeably larger than $s(L)$, that is, whether non-linear gates can ever help us compute linear operators more efficiently. However, we also cannot rule out the possibility that *all* linear operators L are computable by depth-2, non-linear circuits of size $O(n \cdot \text{polylog}(n))$; see [JS10]. (We will at least prove, in Section 4.7 of Chapter 4, that $s(L) = \Omega(n \ln n)$ for random L.)

For several decades, the best known lower bounds for explicit operators (including cyclic convolution) were based on the *superconcentrator technique* of [Val76, Val77]. In a recent breakthrough, Cherukhin [Che08a], obtained new lower bounds giving a (modest) asymptotic improvement over these previous results. (He proved an $\Omega(n^{3/2})$ lower bound for depth 2, and extended the previous lower bounds for any constant depth $d \geq 2$ to apply to circuits of depth $d+1$.) Cherukhin's method, developed specifically for the convolution operator, was later formulated by Jukna [Juk12, Chap. 13] as a general property of operators, called the *Strong Multiscale Entropy (SME)* property, that yields a lower bound of form $\Omega_d(n \cdot \lambda_{d-1}(n))$. Despite the modest nature of the gain over the previous superconcentrator bounds, it was exciting to see *any* progress in this area.

[8] A lower-bound criterion specific to linear circuits, based on *matrix rigidity*, has been given by Valiant [Val77]. In principle this method is capable of showing strong lower bounds. However, except for some limited success in depth 2 [Pud94], no one has proved sufficiently-strong rigidity lower bounds on explicit $\mathbb{F}_2$-matrices to imply circuit lower bounds in this way. See [Lok09] for a survey of this line of work.

1.5 Other work in joint computation

Here we briefly describe three avenues of research that are relevant to joint computation, but do not fit neatly into the areas described in the previous sections. The first topic, *instance compression*, will be our focus of study in Chapter 5, and we will review it in greater detail in that chapter. The other topics we describe here will not be directly studied in this work, but help to provide a fuller sense of joint computation's role in contemporary research.

1.5.1 Joint compression of problem instances

An *instance compression scheme* for a decision problem L is a many-to-one reduction R from L to some second, "target" decision problem L'. That is, R satisfies

$$\overline{x} \in L \iff R(\overline{x}) \in L' \ .$$

(For randomized reductions, we may ask for this equivalence to hold with high probability.) There is no requirement that any *proof* be supplied for the equivalence exhibited by R, nor any requirement that $\overline{x}$ be recoverable from $R(\overline{x})$. We are interested in reductions for which significant compression occurs, that is, for which $R(\overline{x})$ is significantly shorter than $\overline{x}$. Studying the power and limits of instance compression involves an intriguing interplay between computational and information-theoretic ideas.

Instance compression schemes in which the source and target languages are equal ($L = L'$), or *kernelization reductions,* form a central technique in the design of *fixed-parameter algorithms* [DF99], and more general instance compression has also shown to have interesting connections to questions in cryptography [HN10]. A related notion of *core-sets* [AHPV05], which is a family of sparsification techniques principally aimed at geometric data, has also proved to be an influential tool in the design of geometric algorithms.

It is unknown whether one can efficiently, significantly compress an arbitrary instance of a natural NP-complete language like SAT, the set of satisfiable Boolean formulas.[9] However, it is conceivable that efficient compression for SAT might become easier, if we were allowed to *jointly* compress *multiple* SAT *instances into a single output string.* This possibility, suggested by [HN10, BDFH09], can be viewed as an analogy to the direct sum problem in the setting of instance compression.

As an example to make this possibility more concrete, consider the case in which we are given Boolean formulas $\psi_1, \ldots, \psi_{n^{100}}$, each of length n, and want to know whether at least one of them is satisfiable. Can we efficiently compress this question to an easier-to-state, equivalent question about the satisfiability of a single formula $\Phi = R(\psi_1, \ldots, \psi_{n^{100}})$, where Φ is of length, say, n^3?

This kind of "OR-compression" for SAT would be a dramatic counterexample to the "disjoint-inputs intuition" in this setting, which suggests that joint compression of the ψ_j shouldn't yield savings since these instances are "unrelated." It would also go against our intuition that SAT is a formidably hard problem. However, the possibility of this kind of compression is consistent, as far as we know, with the hypothesis $\mathsf{P} \neq \mathsf{NP}$.

Why study the "joint compressibility" of SAT, beyond its resonance with the theme of joint computation? It was shown in [HN10, BDFH09] that this question is intimately connected to open questions about *individual* instance compression. As we describe in Chapter 5, if the "OR-compression" task for SAT (as sketched above) is indeed intractable, then this would imply the intractability of individual instance compression for many natural problems. Hardness of the corresponding "AND-compression" task for SAT would imply the intractability of a number of other problems.

Motivated by these findings, Fortnow and Santhanam [FS11] provided the first strong complexity-theoretic evidence against joint compression for SAT. They showed that a deterministic OR-compression reduction for SAT that reduces $t = t(n) =$

[9]If we could efficiently reduce instances of SAT to shorter instances of SAT itself, then we could iterate the reduction to solve our problem in polynomial time, implying $\mathsf{P} = \mathsf{NP}$. However, even if $\mathsf{P} \neq \mathsf{NP}$, it is still conceivable that SAT might have an efficient compressive reduction to a different target problem—to the Halting problem, say.

poly(n) length-n instances to an output instance of size $O(t \log t)$, of any target problem, would imply the collapse $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. (Their techniques also handle randomized compression reductions that avoid false negatives, with a somewhat more involved statement in this case. Their techniques do not apply to AND-compression for SAT, which remained mysterious after their work.)

Even assuming $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, Fortnow and Santhanam's result does not fully rule out significant deterministic compression for an OR of SAT instances; it says only that the "amortized" output length per input formula cannot be $O(\log n)$. This still provides very useful negative evidence about the feasibility of various instance compression tasks, however, and has helped guide algorithmic research in the area.

1.5.2 Parallel repetition theorems for 2-prover games

A *2-prover (1-round) game* G involves one party called *Verifier*, interacting with two parties, called *Provers 1 and 2*, that cannot communicate directly with one another. Verifier uses a source of randomness to generate two *challenge* strings (w_1, w_2) according to a distribution $\mathcal{D}$ over a finite set of possible messages; $\mathcal{D}$ may involve some dependence between the w_is. Verifier sends w_i to Prover i $(i = 1, 2)$. Each Prover i returns a *response* string z_i. Verifier applies some predicate $P(w_1, w_2, z_1, z_2)$ to decide whether to accept or reject. The *value* $\mathrm{Val}(G) \in [0, 1]$ of the game, defined with respect to P and $\mathcal{D}$, is the maximum achievable success probability of any non-communicating Prover strategy in causing Verifier to accept.

In the *k-fold parallel repetition of G,* denoted $G^{\otimes k}$, Verifier plays k copies of this game simultaneously, with k challenge-pairs drawn independently from $\mathcal{D}$; a single pair of Provers 1 and 2 plays each of the k copies. The Provers' goal is now to make Verifier accept on *all* of the k copies. The basic question is whether the Provers can do significantly better than by playing an optimal independent strategy on each copy. That is, can $\mathrm{Val}\left(G^{\otimes k}\right)$ be significantly larger than $\mathrm{Val}(G)^k$?[10]

This question is not computational in nature, since no computational restrictions

[10]For motivation, in the simpler setting of *1-prover* games, it is not hard to show that $\mathrm{Val}\left(G^{\otimes k}\right) = \mathrm{Val}(G)^k$.

are placed on the Provers. However, it has much of the flavor of the direct product problem for computing multiple independent copies of a function f; ideas and techniques have flowed back and forth between these two areas. It turns out that $\mathrm{Val}\left(G^{\otimes k}\right)$ can indeed be larger than $\mathrm{Val}(G)^k$ [For89], but still exhibits a weaker form of exponential decay. The most important result of this type, the Parallel Repetition Theorem (PRT) of Raz [Raz98], has been a key tool in the study of hardness of approximation for NP-hard problems. See [AB09] for a description of this connection. New variants of the PRT are still being studied and proved; see, e.g., [RR12] for a recent example and overview.

1.5.3 Reductions and equivalences between operators and decision problems

Reductions between problems are a well-established tool for providing evidence of their relative easiness or difficulty. This is true, not only of decision problems, but of multiple-output problems as well. Moreover, reductions can in some cases shed light on the *fine-grained* complexity of a problem, with close attention to polynomial factors; this is important for the study of joint computation, since the efficiency savings from efficient joint computation are in most cases polynomial at best.

The well-studied *All Pairs Shortest Path (APSP)* problem asks for the distances between all pairs of nodes in an n-vertex weighted, directed graph (with weights given by $O(\log n)$-bit integers). A famous open question is whether this problem possesses a *strongly sub-cubic algorithm*, i.e., one whose running time is at most $n^{3-\Omega(1)}$. (Note, here the input length is $\Theta(n^2 \log n)$.)

Vassilevska Williams and Williams [WW10] show that this problem possesses a strongly sub-cubic algorithm *if and only if* a whole list of other fairly natural problems do.[11] One such equivalence is particularly novel and especially interesting from our perspective: the authors show that the APSP problem has a strongly sub-cubic algorithm if and only if a particular *decision problem* has a strongly sub-cubic algo-

[11]See [WW10] for references to some other work on reductions showing tight connections of this kind.

rithm: namely, the problem of detecting a triangle of negative total edge-weight in a weighted n-vertex graph.

The reduction in [WW10] from APSP to this decision problem is rather non-standard, and involves multiple adaptive calls to negative-triangle-detection instances of varying sizes. The authors give another, related example of this kind of "multiple-output to decision-problem equivalence," for the so-called *3-Sum* problem. It seems quite interesting to ask in what generality these kinds of equivalences can be found: which questions about the complexity of efficient joint computation can be converted into equivalent or nearly-equivalent questions about the individual complexity of decision problems?

1.5.4 The Baur-Strassen theorem

In the model of *arithmetic circuits* computing polynomials, Baur and Strassen [BS83] discovered a powerful connection between individual polynomials on the one hand, and collections of polynomials on the other. They showed that over any field, any arithmetic circuit computing a polynomial p can be converted into one that computes p along with *all* of its (formal) partial derivates $\frac{\partial p}{\partial x_i}$ with respect to each input variable x_i; this transformation increases the circuit size by only a constant factor. This is an algorithmic result, but it also has an important corollary for complexity theory: we can transfer lower bounds proved for any *collection* $\mathcal{C}$ of polynomials, to any *single* polynomial p whose set of partial derivatives contains $\mathcal{C}$. Using this connection, Baur and Strassen extended an earlier, tight $\Omega(n \log d)$ lower bound on arithmetic circuit size, proved by Strassen [Str73a] for the polynomial collection $(x_1^d, \ldots, x_n^d)$, to a tight $\Omega(n \log d)$ lower bound for the single polynomial $p = x_1^d + \ldots + x_n^d$. This remains the largest lower bound proved for an explicit degree-d polynomial in the unrestricted arithmetic circuit model. Using their theorem, Baur and Strassen also proved that the determinant polynomial has arithmetic circuit complexity asymptotically equal to that of matrix multiplication.

The Baur-Strassen theorem has had several other applications in proving arithmetic circuit lower bounds, e.g., in [NW95, RS03]. On the algorithmic side, Cygan et al. [CGS12] recently found new and unexpected applications of the Baur-Strassen theorem to graph-theoretic problems.

1.6 Our contributions

In this thesis, we make contributions to four distinct areas of research on joint computation. Our contributions span the query, circuit, and Turing machine models. Here we briefly describe our main contributions; these are explained in greater detail in the opening sections of subsequent chapters. Our results are of several kinds: they variously strengthen our state of knowledge of the limits of efficient joint computation; point to *barriers* to improving known lower bounds for multiple-output tasks; and initiate an entirely new direction for the study of synergies in joint computation.

1.6.1 Improved direct product theorems for randomized query complexity

In Chapter 2, we give a decisive improvement for the known direct product theorems for classical query algorithms, in both the distributional and worst-case error models. Our results establish, for example, that if a Boolean function f cannot be computed with success probability greater than $1 - \varepsilon$ using T queries, then the k-fold direct product $f^{\otimes k}$ cannot be computed with success probability greater than $(1 - 1.1\varepsilon)^k$ by an algorithm using $.1\varepsilon Tk$ queries. Using examples due to Shaltiel [Sha03], we show that the tradeoff established in our result between the query bound we impose and the success probability bound we guarantee is essentially optimal. This is the first fully satisfactory direct product theorem for *any* natural computational model; prior results either imposed significant restrictions on the behavior of algorithms to compute the direct product $f^{\otimes k}$, or else placed resource bounds that in some cases scale as significantly less than k times the resources needed to compute a single instance of f.

We give numerous generalizations of this result, in which we consider more general forms of computational tasks. For example, we study the query complexity of computing (non-Boolean) relations, of computing the k-fold XOR $f^{\oplus k}$ of a Boolean function, and of interacting successfully with dynamic, stateful entities rather than fixed input strings. We also prove so-called "threshold" direct product theorems, in which we upper-bound the probability that a query-bounded algorithm for algorithm to compute $f^{\otimes k}$ even solves f correctly on "too many" of the k input instances. Deterministic query algorithms can be modeled as decision trees, and we are also able to prove a direct product theorem in which decision tree *size* is the resource of interest.

1.6.2 A universality result for joint complexity in the decision tree model

In Chapter 3, we study the following general question about joint computation:

How can we characterize the "diversity" of possible behaviors of the joint complexity of a finite collection $f_1, \ldots, f_k$ of finite Boolean functions defined over a shared input $\overline{x} \in \{0,1\}^n$?

Here, $f_1, \ldots, f_k$ can be chosen arbitrarily; we are interested in what sorts of joint complexity "profiles" can be *realized* by some choice of $f_1, \ldots, f_k$. This question makes sense in various computational models, with respect to various measures of computational cost. Our focus in Chapter 3 will be on computing (total) Boolean functions in the deterministic query model, with an algorithm's cost measured by the worst-case number of queries.

As a representative example of this general question, we may ask:

Are there functions f_1, f_2, f_3, such that: (i) any two f_is have extremely strong "synergies" between them; yet, (ii) the full collection f_1, f_2, f_3 allows only modest gains from joint computation?

This question can be posed more concretely, as follows:

Are there f_1, f_2, f_3, such that: (i) each pair of functions can be jointly computed on a shared input with at most 1.01 *times the resources of computing f_1; yet, (ii) computing f_1, f_2, f_3 jointly requires* 1.99 *times the resources of computing f_1?*

How does one begin to investigate this question? In most "reasonable" models of computation, one has some simple facts about joint complexity. First, the complexity of computing any collection of functions is of course non-negative. Second, the complexity of computing some collection F of functions is at most the complexity of computing the collection F', whenever F' is a superset of F. Third, the complexity of computing two collections F, G jointly is at most the sum of the complexities of computing each of F and G.

As our main result of Chapter 3 (Theorem 3.0.1), we show that, in the query model, these three "obvious" constraints on the behavior of the joint complexity are, in a certain strong sense, *the only ones.* Namely, if we are given some purported "profile" of the complexity of jointly computing each subset of a finite collection of total Boolean functions, and if this profile obeys the three "sanity checks" listed above, then the profile *essentially describes* a valid collection of Boolean functions. The only caveats are that we must be allowed to "scale up" the complexity profile by a scalar multiple of our choice, and we must accept a small $(1 \pm \varepsilon)$ multiplicative error in the predicted joint complexity of each of the various subsets of computational tasks. (We are free to choose any $\varepsilon > 0$. The result holds not just for collections of three functions f_1, f_2, f_3, but for any constant number.)

This "universality result" establishes that the behavior of the joint complexity in the query model is, in a sense, "maximally diverse." It also allows us to affirmatively answer the concrete question about f_1, f_2, f_3 given above. The existence of total functions with this behavior is not simple to show and, in our opinion, comes as a surprise. In Chapter 3 we also present a conjecture about the communication model that would allow us to extend our universality result to that model.

The result of Chapter 3 is the first of its kind, and requires several interesting ingredients to prove. Notably, we define a new type of cryptographic data structure, and construct it using a known separation due to Savický [Sav02] between deterministic and unambiguous-nondeterminstic query complexity. We hope that this work will help inspire other new ways of studying the complexity of joint computation.

1.6.3 Limitations of lower-bound methods for the wire complexity of Boolean operators

Chapter 4 is motivated by the lack of strong results in circuit complexity, and in particular, the lack of strong known lower bounds on the wires needed in constant-depth, arbitrary-gates circuits to compute explicit multiple-output Boolean operators $F : \{0,1\}^n \to \{0,1\}^n$. As described in Section 1.4.3, this is a powerful model of computation in which non-trivial lower bounds only exist for multiple-output operators, and where the whole challenge is to understand the limits of efficient joint computation. Within this model, we ask whether the analysis of several known lower-bound criteria can be improved: that is, we ask whether the properties of Boolean operators known to imply lower bounds might actually imply stronger lower bounds by a better analysis.

Our main object of study is the Cherukhin-Jukna "Strong Multiscale Entropy" (SME) property [Che08a, Juk12], mentioned in Section 1.4.3 and formally introduced in Chapter 4. We also study two simpler lower-bound methods due to Jukna: the "entropy method" for general Boolean operators and the "pairwise-distance method" for $\mathbb{F}_2$-linear operators. The message of our work is that previous analyses of these three methods cannot be significantly improved. To show this, we construct explicit operators that obey these properties, yet which are "easy to compute" in the appropriate sense. Our most important result along these lines is that there is an explicit operator with the SME property, that is computable in depth d with $O(n \cdot \lambda_{d-1}(n))$ wires, for $d = 2, 3$ and for even $d \geq 6$; this matches the Cherukhin-Jukna lower bounds for these depths, up to a constant multiplicative factor depending on d. (See Section 4.4.2 for the definition of the $\lambda_d(\cdot)$ functions.) This identifies an inherent weakness in the best currently-known lower-bound criterion for arbitrary-gates circuits. The techniques in our circuit construction for this main result bear some resemblance to known efficient constructions of bounded-depth superconcentrators [DDPW83], but the details are quite different.

1.6.4 New limits to classical and quantum instance compression

In Chapter 5, we study the complexity of "joint compression" for hard problems—in particular the OR- and AND-compression tasks for SAT and other NP-hard decision problems, as described in Section 1.5.1. Here, we manage to significantly extend the negative results of Fortnow and Santhanam [FS11]. We show that strong enough OR- *or* AND-compression for SAT would imply the existence of *non-uniform, statistical zero-knowledge proof systems* for NP and for coNP; this is an even stronger and more unlikely consequence than $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. This gives the first compelling evidence of hardness for AND-compression of SAT, which also implies the first strong hardness results for a whole family of compression tasks identified in [BDFH09]. Such a result was eagerly sought by the fixed-parameter tractable (FPT) algorithms community, and provides substantial new evidence of limits to the "kernelization approach" to FPT algorithm design.

Our techniques are more robust than those of [FS11], and unlike this past work, our results give evidence even against 2-sided error compression reductions. This strengthens our evidence of the intractability of OR-compression for SAT and for another large family of compression tasks identified in [BDFH09, HN10]. To prove all of these results, we exploit the information bottleneck of an instance compression scheme, using a new and non-trivial method to disguise information being fed into a compressive mapping. Namely, we show that for any set $S \subseteq \{0,1\}^n$, parameters t, t' satisfying $t' \leq O(t \log t)$ and $t, t' \leq \mathrm{poly}(n)$, and for any mapping $R : S^t \to \{0,1\}^{\leq t'}$, there exists an input distribution $\mathcal{D}^*$ over S^t for which any $y \in S$ can be randomly "inserted" into a sample $\overline{x} \sim \mathcal{D}^*$, in such a way that the output distribution $R(\overline{x})$ is not too-strongly affected by the insertion. Crucially, such a $\mathcal{D}^*$ can be found that is efficiently sampleable given $\mathrm{poly}(n)$ bits of non-uniform advice. These "disguising distributions" should be of independent interest, and we are optimistic that they will find other applications.

In Chapter 5 we also define a model of *quantum instance compression*, which generalizes standard (classical) instance compression in two ways: (i) the compression reduction is allowed to be an efficient quantum circuit; (ii) the output of the reduction is allowed to be a *quantum state* ρ. (We merely require that the answer to the instance being compressed can be "recovered" by some measurement on ρ depending solely on the state's size; this measurement need not be efficiently performable.) We investigate whether this richer setting for instance compression allows greater potential to compress an AND or OR of SAT instances. We are able to prove a version of our negative results for the quantum setting: we show that sufficiently strong AND- or OR-compression for SAT would imply the existence of *non-uniform, quantum statistical zero-knowledge proof systems* for all of NP and coNP, a conclusion that seems far-fetched. The quantitative bounds in this quantum result are essentially as strong as those we show in the classical setting.

Chapter 2

Improved Direct Product Theorems for Randomized Query Complexity

This chapter studies the *direct product problem*; as we discussed in Section 1.3, this is one approach to exploring the validity of the "disjoint-inputs intuition." Our focus is on the classical query model, which will be formally introduced in Section 2.1; we reviewed known DPTs in the classical and quantum query models in Section 1.3.3.

2.0.5 Results of this chapter

Our first result is the following direct product theorem in the average-case setting:

Theorem 2.0.1. *Suppose f is a Boolean function and μ is a distribution over inputs to f, such that any T-query randomized algorithm has success probability at most $(1-\varepsilon)$ in computing f on an input from μ. Then for $0 < \alpha \leq 1$, any randomized algorithm making $\alpha\varepsilon Tk$ queries has success probability at most $\left(2^{\alpha\varepsilon}(1-\varepsilon)\right)^k < (1-\varepsilon + .84\alpha\varepsilon)^k$ in computing $f^{\otimes k}$ correctly on k inputs drawn independently from μ.*

We use Shaltiel's examples to show that the tradeoff in Theorem 2.0.1 between the query bound and the error probability is essentially best-possible, at least for general functions f and for small values $\alpha < .01$. (For *specific* functions, the success probability will in some cases decay exponentially even when the number of queries allowed scales as Tk rather than εTk.) Theorem 2.0.1 reveals that small values of ε, as used in Shaltiel's examples, are the only major "obstruction" to strong, general direct product statements in the query model.

Using Theorem 2.0.1, we obtain the following DPT for worst-case error, which strengthens the worst-case DPT of [KŠdW07] mentioned earlier:

Theorem 2.0.2. *For any Boolean function f and $0 < \gamma < 1/4$, any randomized algorithm making at most $\gamma^3 R_2(f)k/11$ queries has worst-case success probability less than $(1/2 + \gamma)^k$ in computing $f^{\otimes k}$ correctly.*

It seems intuitive that some statement like Theorem 2.0.2 should hold, and proving such a DPT was arguably one of the major open problems in classical query complexity.[1]

We also prove a new XOR lemma. Let $B_{k,p}$ denote the binomial distribution on k trials with success probability p.

Theorem 2.0.3. *Suppose that any T-query randomized algorithm has success probability at most $(1 - \varepsilon)$ in computing the Boolean function f on an input from μ. Then for $0 < \alpha \leq 1$, any randomized algorithm making $\alpha\varepsilon Tk$ queries and attempting to compute $f^{\oplus k}$ on k inputs drawn independently from μ has success probability at most*

$$\frac{1}{2}\left(1 + \Pr_{Y \sim B_{k,1-2\varepsilon}}[Y > (1 - \alpha\varepsilon)k]\right),$$

which is less than $\frac{1}{2}\left(1 + [1 - 2\varepsilon + 6\alpha \ln(2/\alpha)\varepsilon]^k\right)$.

Compare the probability bound above with the success probability $\frac{1}{2}(1 + (1 - 2\varepsilon)^k)$, which can be attained using Tk queries by attempting to solve each instance independently and outputting the parity of the guessed bits. The concrete estimate given in Theorem 2.0.3 is meant to illustrate how our bound approaches this value as

[1]While classical query algorithms can be viewed as a subclass of quantum query algorithms, we note that Theorem 2.0.2 is incomparable to the more-recent quantum DPT proved by Lee and Roland [LR12], and mentioned in Section 1.3.3: our result shows exponentially-decaying success probability for a more restricted class of algorithms, but under a potentially larger query bound.

$\alpha \to 0$. By a more careful use of Chernoff inequalities, one can get somewhat tighter bounds for specific ranges of α, ε. An XOR lemma for the worst-case setting can also be derived from our result.

In addition to our "ordinary" DPT (Theorem 2.0.1), we also prove a "threshold" DPT, which bounds the probability that a query-bounded algorithm for $f^{\otimes k}$ solves "many" of the k instances correctly. As one special case, we prove:

Theorem 2.0.4. *Let f be a (not necessarily Boolean) function such that any T-query algorithm has success probability at most $1 - \varepsilon$ in computing f on an input from μ. Fix $\eta, \alpha \in (0, 1]$. Consider any randomized algorithm $\mathcal{R}$ making at most $\alpha\varepsilon Tk$ queries on k independent inputs from μ. The probability that $\mathcal{R}$ computes f correctly on at least ηk of the inputs is at most*

$$\Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (\eta - \alpha\varepsilon)k].$$

Using Chernoff inequalities, Theorem 2.0.4 gives success bounds which decay exponentially in k for any fixed $\alpha, \varepsilon, \eta$, provided $\eta > 1 - \varepsilon + \alpha\varepsilon$. As we will explain, Shaltiel's examples show that this cutoff is nearly best-possible. By setting $\eta := 1$ in Theorem 2.0.4, we also get an ordinary DPT for non-Boolean functions, which for typical parameter settings is stronger than the DPT we'd obtain by a straightforward generalization of our techniques for Theorem 2.0.1. This is the simplest way we know to get such a DPT.

Threshold DPTs have been proved for a variety of models, including, recently, for arbitrary Boolean functions in the quantum query model [LR12]. Unger [Ung09] showed how to derive threshold DPTs from XOR lemmas, and recent work of Impagliazzo and Kabanets [IK10] gave a way to derive threshold DPTs from sufficiently strong DPTs; see also the earlier works cited in [Ung09, IK10]. However, the results of [IK10] do not apply for our purposes, and the threshold DPT we prove is more general than we'd get by applying the results of [Ung09] to our XOR lemma. In any case the proof of our threshold DPT is, we feel, quite natural, and actually forms the basis for the proof of our XOR lemma. Our method for proving threshold DPTs applies to very general threshold events: we give bounds on the probability that the set $S \subseteq [k]$ of instances solved correctly by a query-bounded algorithm is "large," in a sense specified by an arbitrary monotone collection $\mathcal{A}$ of subsets of $[k]$. Generalized threshold DPTs of this form were shown recently by Holenstein and Schoenebeck [HS11] in the circuit model, for a rich class of computational tasks called "weakly verifiable puzzles;" as usual in the circuit model, these DPTs require T' to shrink with k. Our techniques appear unrelated to theirs.

We also prove new DPTs for relations (for which direct *sum* theorems were proved recently by [JKS10]), learning tasks, search problems, and errorless heuristics. Deterministic query algorithms can be equivalently viewed as *decision trees*, and we also prove a DPT for decision trees in which decision tree *size*, rather than depth (i.e., number of queries), is the resource of interest. Impagliazzo, Raz, and Wigderson [IRW94] gave a DPT for decision tree size with "ideal" success probability decay $p' = p^k$, but in the case where the size is not allowed to scale with k, i.e., the setting $T' = T$. By contrast, in our DPT, the success probability decays as $p^{\Omega(k)} = (1-\varepsilon)^{\Omega(k)}$, while the size bound T' scales as $T^{\Omega(\varepsilon k)}$.

Finally, we give a further generalization of our DPTs, in which the k objects being queried are *dynamic entities* rather than static strings—that is, the answers to current queries may depend on past queries. DPTs for dynamic interaction have been proved before [MPR07], but only for the case in which the number of queries to each entity is fixed in advance. (This is analogous to Shaltiel's result for "fair" algorithms.) We further discuss the relation to past work on dynamic interaction in Section 2.9.

In order to ease notation, in this chapter we discuss only DPTs for total functions, but our results apply to partial functions, that is, functions with a restricted domain; the proofs are the same. Similarly, our theorems and proofs carry over without change to handle non-Boolean input alphabets, as well as heterogeneous query costs. Taken as a whole, our results provide a fairly complete picture of the "direct product phenomenon" for randomized query complexity, although there may still be room for improvement in some of our bounds. We hope this work may also help lead to a better understanding of the direct product problem in other, richer computational models.

2.0.6 Our methods

We first explain our method to prove our "basic" direct product theorem, Theorem 2.0.1. As mentioned earlier, Shaltiel [Sha03] proved an optimal DPT for "fair" decision trees, in which each of the k inputs receives T queries. Our proof method for Theorem 2.0.1 also yields an alternate proof of Shaltiel's result, and it is helpful to sketch how this works first. (Really, this "alternate proof" is little more than a rephrasing of Shaltiel's proof technique, but the rephrasing gives a useful perspective which helps us to prove our new results.)

Suppose that every T-query algorithm for computing f succeeds with probability at most $1-\varepsilon$ on an input from the distribution μ. Consider a fair Tk-query algorithm $\mathcal{D}$ for $f^{\otimes k}$, running on k independent inputs from μ. We think of the algorithm as a "gambler" who bets at k "tables," and we define a random variable $X_{j,t} \in [1/2, 1]$ which represents the gambler's "fortune" at the j-th table after $\mathcal{D}$ has made t queries overall to the k inputs. Roughly speaking, $X_{j,t}$ measures how well the algorithm is doing in determining the value of f on the j-th input. When $\mathcal{D}$ queries the j-th input, the j-th fortune may rise or fall, according to the bit seen; we regard each bit revealed to be generated sequentially at random, conditioned on the bits queried so far. The fortunes are defined so that $X_{j,0} \leq 1-\varepsilon$ for each j (reflecting the assumed hardness of f on μ), and so that no action by the algorithm leads to an expected gain in fortune.[2] It follows that $\mathbb{E}[\prod_{j\in[k]} X_{j,Tk}] \leq (1-\varepsilon)^k$. But the fortunes are defined so that $\mathbb{E}[\prod_{j\in[k]} X_{j,Tk}]$ upper-bounds the success probability of $\mathcal{D}$ in computing $f^{\otimes k}$. This gives the DPT for fair algorithms. A key fact underlying the success of this proof strategy is that, after conditioning on any initial sequence of outcomes to the first $t \leq T$ queries by the algorithm, the k inputs remain independent.

If $\mathcal{D}$ is no longer required to be fair, but instead makes at most $\alpha\varepsilon Tk$ queries, then the individual fortune $X_{j,t}$ we define no longer has the same intuitive meaning

[2]In standard probabilistic terms, each individual sequence $X_{j,0}, X_{j,1}, \ldots$ is a *supermartingale*. We will not use this terminology in the present work.

after the j-th input has been queried more than T times. (In this event we simply set $X_{j,t}$ to $1/2$, so that the gambler cannot hope to increase the j-th fortune.) However, the success probability of $\mathcal{D}$ can still be upper-bounded by $\mathbb{E}[\prod_{j \in S} X_{j,\alpha\varepsilon Tk}]$, where S is the (random) set of inputs which receive at most T queries. Counting tells us that fewer than $\alpha\varepsilon k$ of the inputs can lie outside of S, and each fortune is always at least $1/2$, so the success probability is at most $2^{\alpha\varepsilon k}\mathbb{E}[\prod_{j\in[k]} X_{j,\alpha\varepsilon Tk}] \leq 2^{\alpha\varepsilon k}(1-\varepsilon)^k$, giving the statement of Theorem 2.0.1.

Our worst-case DPT for Boolean functions follows straightforwardly from Theorem 2.0.1, by an application of Yao's minimax principle. Our DPT for decision tree size requires a somewhat different analysis, in which we track the "size-usage" of each of the k inputs rather than their number of queries, but the basic approach is the same as in Theorem 2.0.1. In generalizing our method to prove our other results, however, we face a new wrinkle: the natural definitions of the "fortunes" $X_{j,t}$ in these settings are no longer bounded from below by $1/2$. For example, if $f : \{0,1\}^n \to B$ then we have $X_{j,t} \geq |B|^{-1}$, and a straightforward modification of the method described above gives a DPT whose strength degrades as $|B|$ grows. In other settings (e.g., the k-fold XOR setting), we will only have $X_{j,t} \geq 0$, and the method fails completely.[3]

To overcome this difficulty, we adopt a more general perspective. Our previous proof hinged on the fact that, if a gambler plays neutral or unfavorable games at k tables with an initial (nontransferable) endowment of $1-\varepsilon$ at each table, then the probability he reaches a fortune of 1 at every table is at most $(1-\varepsilon)^k$. Note, this is just the success probability he would achieve if he followed an independent "all-or-nothing bet" strategy at each table. It is natural to wonder whether this strategy remains optimal if the gambler wants merely to reach a fortune of 1 at "sufficiently many" tables. Indeed, we prove (by an induction on the number of rounds of gambling) that this is true, where the meaning of "sufficiently many" can be specified by any monotone collection of subsets of $[k]$. Most of our generalizations of Theorem 2.0.1, as well as our XOR lemma, follow readily from this handy "gambling lemma," although

[3]One way to work around the problem is to simply add a small "buffer term" to the fortunes $X_{j,t}$. However, this leads to poorer bounds, and does not yield our generalized threshold DPTs.

care is required to define the correct fortunes in each case.

2.0.7 Organization of the chapter

In Section 2.1 we review preliminaries that are used throughout the chapter and that are needed to state and prove our "basic" DPTs, Theorems 2.0.1 and 2.0.2. We will introduce other definitions as needed in later sections. In Section 2.2 we prove Theorem 2.0.1, and in Section 2.3 we use Shaltiel's examples to analyze the tightness of this result. We prove Theorem 2.0.2 in Section 2.4.

In Section 2.5 we prove our "gambling lemma" (Lemma 2.5.1), and use it to prove a generalized threshold DPT for relations. Theorem 2.0.4 will follow as a special case. We also explain how our threshold DPT implies a DPT for the query complexity of certain learning tasks. We prove Theorem 2.0.3, our XOR lemma, in Section 2.6 (also using Lemma 2.5.1). We define search problems and errorless heuristics in Section 2.7, and give DPTs for these settings.

We prove our DPT for decision tree size in Section 2.8. In Section 2.9, we describe generalizations of our DPTs to settings involving interaction with dynamic entities. We end with some questions for future work.

2.1 Preliminaries

All of our random variables will be defined over finite probability spaces. We let $\mathrm{supp}(X)$ denote the support of a random variable X, i.e., the set of values with nonzero probability. Let $\mu^{\otimes k}$ denote k independent copies of distribution μ.

2.1.1 Randomized decision trees and query complexity

A *decision tree* $\mathcal{D}$ over $\{0,1\}^n$ is a rooted, full binary tree (i.e., each node has either 0 or 2 children), in which interior vertices v are labeled by indices $\mathrm{ind}(v) \in [n]$ and leaf vertices are labeled by values $\ell(v)$ in some finite set B (often $B = \{0,1\}$). The *height* of $\mathcal{D}$ is the length of the longest descending path in $\mathcal{D}$. $\mathcal{D}$ defines a function

$f_{\mathcal{D}} : \{0,1\}^n \to B$ in the following way. On input x we start at the root and follow a descending path through $\mathcal{D}$; at interior node v, we pass to the left subchild of v if $x_{ind(v)} = 0$, otherwise we pass to the right subchild of v. When we reach a leaf vertex v, we output the value $\ell(v)$. Any deterministic algorithm to compute f which queries at most t bits of x on any input can be modeled as a height-t decision tree, and we will freely refer to such a tree as a "t-query deterministic algorithm."

A *randomized decision tree* is a probability distribution $\mathcal{R}$ over deterministic decision trees. Upon receiving the input x, the algorithm samples $\mathcal{D} \sim \mathcal{R}$, then outputs $\mathcal{D}(x)$. (Every randomized query algorithm can be modeled in this fashion.) We write $\mathcal{R}(x)$ to denote the random variable giving the output of $\mathcal{R}$ on input x. We say that $\mathcal{R}$ is a *t-query randomized decision tree* if every decision tree in the support of $\mathcal{R}$ has height at most t.

For $\varepsilon \in [0,1]$ and a function f (not necessarily Boolean), we say that $\mathcal{R}$ *ε-computes* f if for all inputs x, $\Pr[\mathcal{R}(x) = f(x)] \geq 1 - \varepsilon$. Similarly, if μ is a distribution over inputs $x \in \{0,1\}^n$, we say that $\mathcal{R}$ *ε-computes f with respect to μ* if $\Pr_{x\sim\mu}[\mathcal{R}(x) = f(x)] \geq 1 - \varepsilon$, where the probability is taken over the random sample $x \sim \mu$ and the randomness used by $\mathcal{R}$.

For a function $f : \{0,1\}^n \to B$, we define $R_2(f)$, the *two-sided-error randomized query complexity of f*, as the minimum t for which there exists a t-query randomized decision tree which 1/3-computes f. We define

$$\mathrm{Suc}_{T,\mu}(f) \;:=\; 1 - \varepsilon,$$

where $\varepsilon \geq 0$ is the minimum value for which some T-query-bounded randomized algorithm $\mathcal{R}$ ε-computes f with respect to μ. By standard arguments, this minimum exists, and is attained by a deterministic height-T decision tree.

For $f : \{0,1\}^n \to B$ and $k \geq 1$, define $f^{\otimes k} : \{0,1\}^{kn} \to B^k$, the *$k$-fold direct product of f*, as $f^{\otimes k}(x^1, \ldots, x^k) := (f(x^1), \ldots, f(x^k))$. If f is Boolean, define the *k-fold XOR of f* as $f^{\oplus k}(x^1, \ldots, x^k) := f(x^1) \oplus \ldots \oplus f(x^k)$, where $\oplus$ denotes addition mod 2.

2.1.2 Binomial distributions and Chernoff bounds

Let $B_{k,p}$ denote the binomial distribution on k trials with bias p. That is, $B_{k,p}$ is distributed as $Y = \sum_{i=1}^{k} Y_i$, where the Y_i are independent and 0/1-valued with $\Pr[Y_i = 1] = p$. For $s \in \{0, 1, \dots, k\}$ we have the explicit formula $\Pr[Y = s] = \binom{k}{s} p^s (1-p)^{k-s}$.

The following is a general form of Chernoff's inequality:

Lemma 2.1.1 ([DP09], §1.3)**.** *Suppose $Y \sim B_{k,p}$, with $q := 1-p$. Then for $t \in [0, q)$,*

$$\Pr\left[Y > (p+t)k\right] \;\le\; \left(\left(\frac{p}{p+t}\right)^{p+t}\left(\frac{q}{q-t}\right)^{q-t}\right)^{k}.$$

The following form of Chernoff's inequality will be more convenient for us.

Lemma 2.1.2. *Let $\delta \in (0,1)$, and let $Y \sim B_{k,1-\delta}$. If $\beta \in (0, 1/2]$, then*

$$\Pr[Y > (1-\beta\delta)k] \;<\; \left[1 - \delta + 6\beta\ln(1/\beta)\delta\right]^{k}.$$

Proof. We apply Lemma 2.1.1 with $t := (1-\beta)\delta$; we find

$$\begin{aligned}
\Pr\left[Y > (1-\beta\delta)k\right] &= \Pr[Y > ((1-\delta) + (1-\beta)\delta)k] \\
&\le \left(\left(\frac{1-\delta}{1-\beta\delta}\right)^{1-\beta\delta}\left(\frac{\delta}{\delta-(1-\beta)\delta}\right)^{\delta-(1-\beta)\delta}\right)^{k} \\
&= \left(\left(\frac{1-\delta}{1-\beta\delta}\right)^{1-\beta\delta}\beta^{-\beta\delta}\right)^{k} \\
&\le \left((1-\delta+2\beta\delta)^{1-\beta\delta}\,\beta^{-\beta\delta}\right)^{k}, \qquad (2.1)
\end{aligned}$$

using $\beta\delta \le 1/2$.

It is easy to verify that $(1-\delta+2\beta\delta) \ge \beta$, so that

$$(1-\delta+2\beta\delta)^{-\beta\delta}\cdot\beta^{-\beta\delta} \;\le\; \beta^{-2\beta\delta} \;=\; e^{2\beta\ln(1/\beta)\delta}.$$

Now $2\beta\ln(1/\beta)\delta \le 2/e < .74$. By convexity of e^x, we have $e^x \le 1 + ((e^{.74}-1)/.74)\cdot x \le$

$1 + 1.49x$ for all $x \in [0, .74]$. Thus, $e^{2\beta \ln(1/\beta)\delta} \leq 1 + 3\beta \ln(1/\beta)\delta$. Combining these facts with Eq. 2.1, we get

$$\begin{aligned} \Pr\left[Y > (1-\beta\delta)k\right] &\leq \left[(1-\delta+2\beta\delta)(1+3\beta\ln(1/\beta)\delta)\right]^k \\ &< \left[1-\delta+6\beta\ln(1/\beta)\delta\right]^k . \end{aligned}$$

□

The constant 6 in Lemma 2.1.2 is not best-possible. To apply the lemma, it is helpful to understand the behavior of the function $h(x) := x\ln(1/x)$. This function is increasing on $(0, e^{-1}]$, and as $x \to 0$, $h(x)$ approaches 0 only slightly more slowly than x itself: for an integer $n > 1$ we have

$$h\left(\frac{1}{2n\ln n}\right) = \frac{1}{2n\ln n} \cdot \ln(2n\ln n) = \frac{1}{n} \cdot \frac{\ln(2n\ln n)}{\ln(n^2)} < \frac{1}{n} .$$

2.2 Proof of Theorem 2.0.1

In this section we prove our "basic" direct product theorem:

Theorem 2.2.1 (Theorem 2.0.1, restated)**.** *Let f be a Boolean function for which* $\mathrm{Suc}_{T,\mu}(f) \leq 1-\varepsilon$. *Then for* $0 < \alpha \leq 1$, $\mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(f^{\otimes k}) \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k < (1-\varepsilon + .84\alpha\varepsilon)^k$.

There is no requirement that T be an integer; this will be useful later in proving Theorem 2.0.2. The success bound $(2^{\alpha\varepsilon}(1-\varepsilon))^k$ above is actually valid for any $\alpha > 0$, but the bound is trivial whenever $\alpha \geq 2$, so we focus attention on a range where the bound is always meaningful.

Proof. The statement is trivial if $T = 0$ or $\varepsilon = 0$, so assume both are positive. By convexity, it is sufficient to show the statement for deterministic algorithms. Also, by a standard limiting argument, it is enough to prove this result under the assumption that $\mathrm{supp}(\mu) = \{0,1\}^n$; this ensures that conditioning on any sequence of query outcomes will be well-defined.

Next we set up some notation and concepts relating to the computation of f on a single input; afterward we will apply our work to the direct-product setting.

For a string $u \in \{0, 1, *\}^n$, let the distribution $\mu^{(u)}$ be defined as a sample from μ, conditioned on the event $[x_i = u_i, \forall i \text{ such that } u_i \in \{0, 1\}]$. Let $|u|$ denote the number of 0/1 entries in u. Let $u[x_i \leftarrow b]$ denote the string u with the i-th coordinate set to b. In our proof we consider the bits of an input $\mathbf{y} \sim \mu$ to be generated sequentially at random as they are queried. Thus if an input is drawn according to μ, and u describes the outcomes of queries made so far (with $*$ in the coordinates that have not been queried), we consider the input to be in the "state" $\mu^{(u)}$. If some index $i \in [n]$ is queried next, then the algorithm sees a 0 with probability $\Pr_{\mathbf{y} \sim \mu^{(u)}}[\mathbf{y}_i = 0]$, in which case the input enters state $\mu^{(u[x_i \leftarrow 0])}$; with the remaining probability the algorithm sees a 1 and the input enters state $\mu^{(u[x_i \leftarrow 1])}$. Clearly this interpretation is statistically equivalent to regarding the input as being drawn from μ before the algorithm begins (this is the "principle of deferred decisions" of probability theory).

For each $u \in \{0, 1, *\}^n$ with $|u| \leq T$, let

$$W(u) \; := \; \mathrm{Suc}_{T-|u|, \mu^{(u)}}(f).$$

In words, $W(u)$ measures our "winning prospects" of computing f on μ, if we begin with a budget of T queries and our first $|u|$ queries reveal the bits described by u, and if we follow an optimal strategy thereafter. Clearly $W(u) \in [1/2, 1]$, since an algorithm may simply guess a random bit. We make two more simple claims about this function.

Lemma 2.2.2. *1.* $W(*^n) \leq 1 - \varepsilon$.

2. For any $u \in \{0, 1, *\}^n$ *with* $|u| < T$, *and any* $i \in [n]$, $\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W(u[x_i \leftarrow \mathbf{y}_i])] \leq W(u)$.

Proof. 1: This is immediate from our initial assumption $\mathrm{Suc}_{T,\mu}(f) \leq 1 - \varepsilon$.

2: If the i-th coordinate has already been queried (i.e., $u_i \in \{0, 1\}$), then $\mathbf{y}_i = u_i$ with probability 1, so $u[x_i \leftarrow \mathbf{y}_i] = u$ and the statement is trivial. So assume $u_i = *$.

Let $\mathcal{R}_0$, $\mathcal{R}_1$ be algorithms making at most $T - (|u| + 1)$ queries and maximizing the success probabilities on $\mu^{(u[x_i \leftarrow 0])}$, $\mu^{(u[x_i \leftarrow 1])}$ respectively. Thus, the success probability of $\mathcal{R}_b$ is $W(u[x_i \leftarrow b])$. Consider an algorithm $\mathcal{R}$ which queries x_i, then runs $\mathcal{R}_b$ if the bit seen is b. $\mathcal{R}$ makes at most $T - |u|$ queries, and the success probability of $\mathcal{R}$ is $\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W(u[x_i \leftarrow \mathbf{y}_i])]$. Thus $W(u)$ is at least this value. $\square$

Now we prove the Theorem. Let $\mathcal{D}$ be any deterministic algorithm making at most $M := \lfloor \alpha \varepsilon T k \rfloor$ queries, and attempting to compute $f^{\otimes k}$ on input strings $(\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$. For $j \in [k]$ and $0 \le t \le M$, let $u_t^j \in \{0, 1, *\}^n$ be the random string giving the outcomes of all queries made to $\mathbf{x}^j$ after $\mathcal{D}$ has made t queries (to the entire input). We need the following simple but important observation:

Lemma 2.2.3. *Condition on any execution of $\mathcal{D}$ for the first $t \ge 0$ steps, with query outcomes given by $u_t^1, \ldots, u_t^k$. Then the input is in the state $\mu^{(u_t^1)} \times \ldots \times \mu^{(u_t^k)}$. That is, the k inputs are independent, with $\mathbf{x}^j$ distributed as $\mu^{(u_t^j)}$.*

Proof. Fix any $j \in [k]$ and consider any assignment $(x^{j'})_{j' \in [k] \setminus \{j\}}$ of values $x^{j'} \in \{0,1\}^n$ to the inputs other than the j-th input, where $x^{j'}$ extends $u_t^{j'}$ for each $j' \neq j$. We show that, after conditioning on the query outcomes $u_t^1, \ldots, u_t^k$ *and* on the event $[\mathbf{x}^{j'} = x^{j'} \ \forall j' \neq j]$, the j-th input $\mathbf{x}^j$ is distributed according to $\mu^{(u_t^j)}$. This will prove the Lemma.

Consider each $y \in \{0,1\}^n$ which extends u_t^j. Now $u_t^1, \ldots, u_t^k$ are, by assumption, a possible description of the first t queries made by $\mathcal{D}$ under *some* input. Since $\mathcal{D}$ is deterministic, and $(x^1, \ldots, x^{j-1}, y, x^{j+1}, \ldots, x^k)$ are consistent with $(u_t^1, \ldots, u_t^k)$, we conclude that $(u_t^1, \ldots, u_t^k)$ also describe the first t queries made by $\mathcal{D}$ on $(x^1, \ldots, x^{j-1}, y, x^{j+1}, \ldots, x^k)$. Thus the conditional probability that $\mathbf{x}^j = y$ is

$$\begin{aligned}
\frac{\mu^{\otimes k}(x^1, \ldots, x^{j-1}, y, x^{j+1}, \ldots, x^k)}{\sum_{z \text{ extends } u_t^j} \mu^{\otimes k}(x^1, \ldots, x^{j-1}, z, x^{j+1}, \ldots, x^k)} &= \\
\frac{\mu(y) \cdot \prod_{j' \neq j} \mu(x^{j'})}{\sum_{z \text{ extends } u_t^j} \mu(z) \cdot \prod_{j' \neq j} \mu(x^{j'})} &= \\
\frac{\mu(y)}{\sum_{z \text{ extends } u_t^j} \mu(z)} &= \mu^{(u_t^j)}(y),
\end{aligned}$$

by definition of $\mu^{(u_t^j)}$. This proves Lemma 2.2.3. $\square$

Next, define collections

$$\mathcal{X} \;=\; \{X_{j,t}\}_{j\in[k],0\le t\le M}, \quad \mathcal{P} \;=\; \{P_t\}_{0\le t\le M}$$

of random variables, as follows. All the random variables are determined by the execution of $\mathcal{D}$ on an input drawn from $\mu^{\otimes k}$. Let $X_{j,t} := W(u_j^t)$ if $|u_j^t| \le T$; otherwise let $X_{j,t} := 1/2$. Let $P_t := \prod_{j\in[k]} X_{j,t}$.

We claim that for each $0 \le t < M$, $\mathbb{E}[P_{t+1}] \le \mathbb{E}[P_t]$. To see this, condition on any outcomes to the first t queries, described by $u_t^1, \ldots, u_t^k$. Now suppose that for the $(t+1)$-st query, $\mathcal{D}$ queries the i-th bit of the j-th input (i, j are determined by $u_t^1, \ldots, u_t^k$, since $\mathcal{D}$ is deterministic). We note that $X_{j',t+1} = X_{j',t}$ for all $j' \ne j$. If $|u_j^t| \ge T$ then also $X_{j,t+1} \le X_{j,t}$, which implies $P_{t+1} \le P_t$. So assume $|u_j^t| < T$. Then we have

$$\mathbb{E}[P_{t+1}|u_t^1,\ldots,u_t^k] \;=\; \mathbb{E}[X_{j,t+1}\cdot\prod_{j'\ne j} X_{j',t+1}|u_t^1,\ldots,u_t^k]$$

$$=\; \mathbb{E}[X_{j,t+1}|u_t^1,\ldots,u_t^k]\cdot\prod_{j'\ne j} X_{j',t} \;\le\; X_{j,t}\cdot\prod_{j'\ne j} X_{j',t} \;=\; P_t,$$

where we used Lemma 2.2.3 and part 2 of Lemma 2.2.2. We conclude

$$\mathbb{E}[P_{t+1}] \;=\; \mathbb{E}[\mathbb{E}[P_{t+1}|u_t^1,\ldots,u_t^k]] \;\le\; \mathbb{E}[P_t],$$

as claimed. It follows that $\mathbb{E}[P_M] \le \mathbb{E}[P_0]$. But we can bound P_0 directly: $P_0 = W(*^n)^k \le (1-\varepsilon)^k$ (Lemma 2.2.2, part 1). Thus $\mathbb{E}[P_M] \le (1-\varepsilon)^k$.

Now we argue that this implies an upper bound on the success probability of $\mathcal{D}$. Condition on the bits $u_M^1, \ldots, u_M^k$ seen by $\mathcal{D}$ during a complete execution; these determine the k output bits of $\mathcal{D}$. For each $j \in [k]$, at least one of two possibilities holds: either $|u_M^j| > T$, or the j-th input is in a final state $\mu^{(u_M^j)}$ for which $\Pr_{\mathbf{y}\sim\mu^{(u_M^j)}}[f(\mathbf{y}) = 1] \in [1 - X_{j,M}, X_{j,M}]$. Since the k inputs remain independent under our conditioning, the conditional probability that $\mathcal{D}$ computes $f^{\otimes k}$ correctly is at most $\prod_{j:|u_M^j|\le T} X_{j,M}$.

$\mathcal{D}$ makes at most $\alpha\varepsilon T k$ queries, so simple counting tells us that there are fewer than $\alpha\varepsilon k$ indices j for which $|u_M^j| > T$. Thus,

$$\prod_{j:|u_M^j|\leq T} X_{j,M} \;\leq\; \frac{\prod_{j\in[k]} X_j}{(\min_{j\in[k]} X_{j,M})^{\alpha\varepsilon k}} \;\leq\; 2^{\alpha\varepsilon k} P_M$$

(since $X_{j,M} \geq 1/2$ for all j). Taking expectations, we find that the *overall* success probability of $\mathcal{D}$ is at most $\mathbb{E}[2^{\alpha\varepsilon k} P_M] \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k$.

Finally, we simplify our bound. We claim $2^x < 1+.84x$ on $(0, 1/2]$. To see this, just note that $2^0 = 1$, that $2^{1/2} < 1.42 = 1 + .84(1/2)$, and that 2^x is a convex function on $\mathbb{R}$. Then, since $0 < \alpha\varepsilon \leq 1/2$, we have $2^{\alpha\varepsilon}(1-\varepsilon) < (1+.84\alpha\varepsilon)(1-\varepsilon) < 1-\varepsilon+.84\alpha\varepsilon$. The proof is complete. $\square$

We remark that, as claimed in Section 2.0.5, the proof above can be easily adapted to give an alternate proof of Shaltiel's optimal direct product theorem for "fair" algorithms making Tk queries: we define the random variables $X_{j,t}$ exactly as before and note that $|u_t^j| \leq T$ for all j, t.

2.3 Tightness of the bounds in Theorem 2.0.1

In this section we describe a family of functions and input distributions, due to [Sha03], and explain why they show that the query/success tradeoff in Theorem 2.0.1 is nearly best-possible, at least when $\alpha < .01$ and when $(1-\varepsilon)^k$ is also at most a small constant.

Fixing an integer $T > 0$, define $f_T : \{0,1\}^{T+2} \to \{0,1\}$ as follows: let $f_T(x) := x_2$ if $x_1 = 1$, otherwise $f_T(x) := x_2 \oplus \ldots \oplus x_{T+2}$. Given $\varepsilon \in (0, 1/2)$, let μ_ε be the distribution over $\{0,1\}^{T+2}$ in which all bits are independent, $\Pr[x_1 = 1] = 1-2\varepsilon$, and $\Pr[x_i = 1] = 1/2$ for all $i \in \{2, \ldots, T+2\}$. Note that if $\mathbf{y} \sim \mu_\varepsilon$, a T-query-bounded algorithm can gain no information about the value of f when $x_1 = 0$, so any such algorithm succeeds with probability at most $(1-2\varepsilon)1 + (2\varepsilon)\frac{1}{2} = 1-\varepsilon$ in computing $f(\mathbf{y})$.

Now consider the following algorithm $\mathcal{D}$ attempting to compute $f^{\otimes k}$ on inputs $(\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu_\varepsilon^{\otimes k}$. First $\mathcal{D}$ queries the first two bits of each input. Call an input $\mathbf{x}^k$ "bad" if its first bit is 0, "good" if its first bit is 1. Let $B \subseteq [k]$ denote the set of bad inputs. Note that $\mathcal{D}$ learns the value of f on each good input. Next, $\mathcal{D}$ chooses arbitrarily a set $S \subseteq B$ of $\lfloor \alpha\varepsilon k \rfloor$ bad inputs, and spends T additional queries on each input in S to determine the value of f on these inputs (if there are fewer than $\lfloor \alpha\varepsilon k \rfloor$ bad inputs, $\mathcal{D}$ queries them all and determines the value of $f^{\otimes k}$ with certainty). Finally, $\mathcal{D}$ outputs the answer bits it has learned and makes random guesses for the remaining values.

Observe that $\mathcal{D}$ uses at most $2k + \alpha\varepsilon T k$ queries overall. To analyze the success probability of $\mathcal{D}$, first consider an algorithm $\mathcal{D}'$ which uses only $2k$ queries to look at the two bits of each input; $\mathcal{D}'$ outputs the correct value on good inputs, and guesses randomly on bad inputs. It is easy to see that $\mathcal{D}'$ succeeds with probability $(1-\varepsilon)^k$ in computing $f^{\otimes k}$. Also, if $\mathcal{D}$ and $\mathcal{D}'$ are both run on a common k-tuple of inputs drawn from $\mu_\varepsilon^{\otimes k}$, and we condition on the event that $|B| \geq \lfloor \alpha\varepsilon k \rfloor$, then the success probability of $\mathcal{D}$ is $2^{\lfloor \alpha\varepsilon k \rfloor}$ times the success probability of $\mathcal{D}'$, since the inputs are independent and $\mathcal{D}$ has $\lfloor \alpha\varepsilon k \rfloor$ fewer random guesses to make. Thus, $\Pr[\mathcal{D} \text{ succeeds}]$ is at least

$$\begin{aligned}
& \Pr[|B| \geq \alpha\varepsilon k] \cdot 2^{\lfloor \alpha\varepsilon k \rfloor} \Pr\left[\mathcal{D}' \text{ succeeds} \middle| |B| \geq \alpha\varepsilon k\right] \\
= \; & 2^{\lfloor \alpha\varepsilon k \rfloor} \Pr[\mathcal{D}' \text{ succeeds} \wedge |B| \geq \alpha\varepsilon k] \\
\geq \; & 2^{\lfloor \alpha\varepsilon k \rfloor} \cdot (\Pr[\mathcal{D}' \text{ succeeds}] - \Pr[|B| < \alpha\varepsilon k]) \\
= \; & 2^{\lfloor \alpha\varepsilon k \rfloor} \cdot \big((1-\varepsilon)^k - \Pr[|B| < \alpha\varepsilon k]\big) . \qquad (2.2)
\end{aligned}$$

Define the indicator variable $Y_j := \mathbf{1}_{[j \notin B]}$; then the Y_j's are independent, with $p = \Pr[Y_j = 1] = 1 - 2\varepsilon$. Let $Y := Y_1 + \ldots + Y_k$. We apply Lemma 2.1.2 to Y, with the settings $\delta := 2\varepsilon$ and $\beta := \alpha/2 \leq 1/2$, to obtain

$$\begin{aligned}
\Pr[|B| < \alpha\varepsilon k] &= \Pr[Y > (1-\alpha\varepsilon)k] \\
&= \Pr[Y > (1-(2\varepsilon)(\alpha/2))k]
\end{aligned}$$

$$< \ [1 - 2\varepsilon + 6(\alpha/2)\ln(2/\alpha)(2\varepsilon)]^k .$$

This can be made less than $(1-1.5\varepsilon)^k$ if α is a small enough positive constant ($\alpha < .01$ will work).

Now if $(1-\varepsilon)^k$ is also at most a sufficiently small constant, then $(1-1.5\varepsilon)^k < .1(1-\varepsilon)^k$ so that, by Eq. 2.2,

$$\Pr[\mathcal{D} \text{ succeeds}] \ > \ .9 \cdot 2^{\lfloor \alpha\varepsilon k \rfloor}(1-\varepsilon)^k,$$

which is close to the maximum success probability allowed by Theorem 2.0.1 if $\mathcal{D}$ used $\alpha\varepsilon T k$ queries. (Recall, though, that $\mathcal{D}$ uses $2k + \alpha\varepsilon T k$ queries.)

2.4 Proof of Theorem 2.0.2

We now prove Theorem 2.0.2 from Section 2.0.5, our DPT for worst-case error, by combining Theorem 2.0.1 with a version of Yao's minimax principle [Yao77], which allows us to convert worst-case hardness assumptions in query complexity into average-case assumptions.

Define $R_{2,\delta}(f)$ as the minimum T for which there exists a randomized T-query algorithm which computes $f(x)$ correctly with probability at least $1-\delta$ for every x. The following is a common version of Yao's principle, and can be proved directly using the minimax theorem of game theory.

Lemma 2.4.1. *Fix $0 < \delta < 1/2$ and a Boolean function f. There exists a distribution μ_δ over inputs to f, such that every randomized algorithm making fewer than $R_{2,\delta}(f)$ queries succeeds in computing f on μ_δ with probability less than $1-\delta$.*

Proof of Theorem 2.0.2. Let f be given. Let $\delta := 1/2 - \gamma/2$, and let $\mu := \mu_\delta$ be as provided by Lemma 2.4.1. Now fix a tiny constant $c \in (0,1)$, and let $T := R_{2,\delta}(f) - c$; we have

$$\mathrm{Suc}_{T,\mu}(f) \ \leq \ 1-\varepsilon,$$

for some value $\varepsilon > \delta > 3/8$ (independent of c). Now set $\alpha := \gamma$, and apply Theorem 2.0.1 to find

$$\mathrm{Suc}_{\gamma\varepsilon Tk,\mu}(f) \; < \; (1-(1-.84\gamma)\varepsilon)^k \; < \; (1-(1-.84\gamma)\delta)^k .$$

Note that $\gamma\varepsilon Tk > \gamma\delta R_{2,\delta}(f)k$, if c is chosen sufficiently small. We conclude that any algorithm making at most $\gamma\delta R_{2,\delta}(f)k$ queries succeeds with probability less than

$$\begin{aligned}(1-(1-.84\gamma)\delta)^k \; &= \; (1-(1-.84\gamma)(1/2-\gamma/2))^k \\ &< \; (1/2+.42\gamma+\gamma/2)^k \; < \; (1/2+\gamma)^k\end{aligned}$$

in computing $f^{\otimes k}$ on inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k \sim \mu^{\otimes k}$. So, the worst-case success probability is also less than this amount.

Now we relate $R_{2,\delta}(f)$ to $R_2(f)$ by standard sampling ideas. Say $\mathcal{R}_\delta$ is an algorithm making $R_{2,\delta}(f)$ queries, which computes $f(x)$ with probability at least $1-\delta = 1/2 + \gamma/2$ on each input. Let $\mathcal{R}$ be the algorithm which given an input x, runs $\mathcal{R}_\delta(x)$ for $m := \lceil 3/\gamma^2 \rceil$ trials, outputting the majority value. For $i \in [m]$, define the indicator variable Y_i for the event [$\mathcal{R}_\delta$ succeeds on the i-th trial], and let $Y := Y_1 + \ldots + Y_m$. Then the probability that $\mathcal{R}(x)$ outputs an incorrect value is at most the probability that $Y \le \mathbb{E}[Y] - \gamma m/2$, which by Hoeffding's inequality is at most $e^{-2\gamma^2 m/4} \le e^{-3/2} < 1/3$.

Thus, $R_2(f) \le R_{2,\delta}(f) \cdot \lceil 3/\gamma^2 \rceil < 4R_{2,\delta}(f)/\gamma^2$ (using $\gamma < 1/4$). Then, we have

$$\gamma^3 R_2(f)k/11 \; < \; \gamma(3/8)(\gamma^2 R_2(f)/4)k \; < \; \gamma\delta R_{2,\delta}(f)k,$$

from which Theorem 2.0.2 follows. □

2.5 Threshold direct product theorems

In this section we prove our "gambling lemma," Lemma 2.5.1, and use it to prove generalized threshold DPTs for relations (relation problems are formally defined in Section 2.5.2). This will yield DPTs for non-Boolean functions as well as for the query complexity of learning tasks. Further applications of Lemma 2.5.1 will appear in later sections.

Let $\mathcal{P}([k])$ denote the collection of subsets of $[k]$. Say that a subcollection $\mathcal{A} \subseteq \mathcal{P}([k])$ is *monotone* if $[A \in \mathcal{A},\ A \subseteq A']$ implies $A' \in \mathcal{A}$. Monotone collections play an important role in what follows.

2.5.1 A gambling lemma

Like the proof of Theorem 2.0.1, the statement of our next lemma is best explained by a gambling metaphor. Suppose that a gambler gambles at k tables, bringing an initial endowment of $p_j \in [0, 1]$ to the j-th table. He cannot transfer funds between tables, or go into debt at any table; he can only play games for which his expected winnings are nonpositive; and the different tables' games use independent randomness. However, the gambler can choose which game to play next at each table.

The gambler wants to reach a fortune of 1 at "sufficiently many" of the tables, where the meaning of "sufficiently many" is specified by a monotone subset $\mathcal{A} \subseteq \mathcal{P}([k])$. One way the gambler may attempt to reach this goal is to simply place an "all-or-nothing" bet independently at each table; that is, at the j-th table, the gambler wins a fortune of 1 with probability p_j, and loses his j-th endowment with the remaining probability. The following lemma states that this is in fact the gambler's best strategy.

Lemma 2.5.1. *Suppose $k, N \geq 1$ are given, along with a collection $\{\mathcal{X}, \mathcal{U}\}$ of random variables (over a finite probability space). Here $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$, where for each $j \in [k]$, $\mathcal{X}_j = \{X_{j,0}, X_{j,1}, \ldots, X_{j,N}\}$ is a sequence of variables in the range $[0, 1]$* (think of $X_{j,t}$ as the gambler's fortune at the j-th table after the first t steps). *$\mathcal{U} = \{U_0, U_1, \ldots, U_{N-1}\}$ is a sequence of random variables taking values over some finite set* (think of U_t as describing the form and outcomes of all gambles in the first t steps). *Assume that for all $0 \leq t < N$, U_t determines $\{X_{1,t}, \ldots, X_{k,t}\}$, and also determines $U_{t'}$ for all $t' < t$. Also assume that $\{X_{1,t+1}, \ldots, X_{k,t+1}\}$ are independent conditioned on U_t. Then, if $X_{j,0} \leq p_j \in [0, 1]$ for all $j \in [k]$, and $\mathcal{A}$ is a monotone* *subset of $\mathcal{P}([k])$, we have*

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{A}] \;\leq\; \Pr[D \in \mathcal{A}],$$

where $D \subseteq [k]$ is generated by independently including each $j \in [k]$ in D with probability p_j.

Note that we assume the gambler never attains a fortune greater than 1 at any table; this restriction is easily removed, but it holds naturally in the settings where we'll apply the Lemma.

Proof. We use the term "$\mathcal{A}$-success" to refer to the event $[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{A}]$ whose probability we are bounding.

We first make a simplifying observation: we claim it is without loss of generality to assume that between each consecutive times $(t, t+1)$, at most one of the fortunes changes, and that the fortune subject to change is determined by t. Call a family of sequences with this property "nice." To see this, consider any family $\mathcal{X}$ obeying Lemma 2.5.1's assumptions, and modify it by "splitting" each transition $(t, t+1)$ into a sequence of k transitions, in the j-th of which the j-th fortune changes (according to the same distribution governing its transition in the original sequence).

More formally, we define $\mathcal{X}'_j = \{X'_{j,0}, \ldots, X'_{j,Nk}\}$ by letting $X'_{j,\ell} := X_{j,\lfloor(\ell+k-j)/k\rfloor}$; and we define $\mathcal{U}' = \{U'_0, U'_1, \ldots, U'_{Nk-1}\}$ by

$$U'_\ell \;:=\; \left(U_{\lfloor \ell/k \rfloor}, \left(X'_{j,\ell'}\right)_{j\in[k], \ell' \leq \ell}\right).$$

(We add extra information into U'_ℓ to ensure that it determines the random variables it is supposed to.) Lemma 2.5.1's assumptions continue to hold for this modified, nice family of random variables; here we are using our original assumption that $\{X_{1,t+1}, \ldots, X_{k,t+1}\}$ are independent conditioned on U_t. Also, the probability of $\mathcal{A}$-success is unchanged. So let us assume from now on that $(\mathcal{X}, \mathcal{U})$ is nice, and for $0 \leq t < N$, let $j_t \in [k]$ be the index of the fortune subject to change between times t and $t+1$.

Fix any $k \geq 1$; we prove the statement by induction on $N \geq 1$. First suppose $N = 1$, and let j_0 be as defined above. Let $S \subseteq [k] \setminus \{j_0\}$ be the set of indices $j \neq j_0$ for which $p_j = 1$. First suppose $S \in \mathcal{A}$; then $\Pr[D \in \mathcal{A}] = 1$, since each $j \in S$ is included in D with probability 1. In this case the conclusion is trivially satisfied. Next suppose $S \cup \{j_0\} \notin \mathcal{A}$. In this case, $\Pr[\mathcal{A}\text{-success}] = 0$, and again the conclusion is trivially satisfied. So suppose $S \notin \mathcal{A}$, $S \cup \{j_0\} \in \mathcal{A}$, and condition on any value $U_0 = u$. Then $\mathcal{A}$-success occurs iff $X_{j_0,1} = 1$. By Markov's inequality, $\Pr[X_{j_0,1} = 1 | U_0 = u] \leq \mathbb{E}[X_{j_0,1} | U_0 = u] \leq X_{j_0,0} \leq p_{j_0} = \Pr[D \in \mathcal{A}]$. This proves the statement for $N = 1$.

So let $N > 1$ and assume the statement proved for $\{1, \ldots, N-1\}$; we prove it for N. Condition on any value $U_0 = u$, and condition further on the value $X_{j_0,1} = a \in [0,1]$. The equalities $X_{j,1} = X_{j,0} \leq p_j$ are forced for all $j \neq j_0$; the residual collection of random variables $\{X_{j,t} : j \in [k], 1 \leq t \leq N\} \cup \{U_t : 1 \leq t < N\}$ under our conditioning obey Lemma 2.5.1's assumptions, along with our added assumption; and these sequences are shorter by a step than our initial sequences. Thus our induction hypothesis implies that

$$\Pr[\mathcal{A}\text{-success} | U_0 = u, X_{j_0,1} = a] \;\leq\; \Pr[D^{(a)} \in \mathcal{A}], \tag{2.3}$$

where $D^{(a)}$ is generated just like D except that j_0 is now included in $D^{(a)}$ with probability a.

Let $q_0 := \Pr[D \setminus \{j_0\} \in \mathcal{A}]$ and $q_1 := \Pr[D \cup \{j_0\} \in \mathcal{A}]$. Note that $q_0 \leq q_1$, since $\mathcal{A}$ is monotone. We have

$$\Pr[D^{(a)} \in \mathcal{A}] \;=\; (1-a)q_0 + aq_1.$$

Taking expectations over a in Eq. 2.3, $\Pr[\mathcal{A}\text{-success} | U_0 = u]$ is at most

$$\begin{aligned} &(1 - \mathbb{E}[X_{j_0,1} | U_0 = u])q_0 + \mathbb{E}[X_{j_0,1} | U_0 = u] \cdot q_1 \\ &\leq\; (1 - p_{j_0})q_0 + p_{j_0} q_1 \end{aligned}$$

$$\begin{aligned} & \text{(since } q_0 \le q_1 \text{ and } \mathbb{E}[X_{j_0,1}|U_0 = u] \le X_{j_0,0} \le p_{j_0}) \\ = \; & \Pr[D \in \mathcal{A}]. \end{aligned}$$

As u was arbitrary, this extends the induction to N, and completes the proof. □

2.5.2 Application to threshold DPTs

Now we prove our generalized threshold direct product theorem. Our theorem will be within the framework of solving relation problems, a more general task than computing functions. A *relation* (with Boolean domain) is a subset $P \subseteq \{0,1\}^n \times B$, for some finite set B. The relation is *total* if for all $x \in \{0,1\}^n$, there exists $b \in B$ such that $(x, b) \in P$. For each total relation P there is a natural computational problem: given an input x, try to output a b for which $(x, b) \in P$. Computing a function $f : \{0,1\}^n \to B$ is equivalent to solving the relation problem for the total relation $P_f := \{(x, b) : f(x) = b\}$.

If $\mathcal{R}$ is a (possibly randomized) query algorithm producing outputs in B, P is a total relation, and μ a distribution, say that $\mathcal{R}$ *ε-solves P with respect to μ* if $\Pr_{x\sim\mu}[(x, \mathcal{R}(x)) \in P] \ge 1-\varepsilon$. Define $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) := 1-\varepsilon$, where $\varepsilon \ge 0$ is the minimum value for which some T-query randomized algorithm $\mathcal{R}$ ε-solves P with respect to μ. As usual, this minimum exists and is attained by a deterministic height-T decision tree. For a randomized algorithm $\mathcal{R}$ making queries to $k \ge 1$ inputs $x = (x^1, \ldots, x^k)$ to P and producing an output in B^k, let $\mathcal{R}_j(x) \in B$ be the j-th value outputted by $\mathcal{R}$.

Given $A, A' \subseteq [k]$, define the *distance* $d(A, A') := |(A \setminus A') \cup (A' \setminus A)|$. Given a set family $\mathcal{A} \subseteq \mathcal{P}([k])$, and a real number $r > 0$, define the *strict r-neighborhood of $\mathcal{A}$*, denoted $N_r(\mathcal{A})$, as

$$N_r(\mathcal{A}) \;:=\; \{A' : d(A, A') < r \text{ for some } A \in \mathcal{A}\}.$$

We have $\mathcal{A} \subseteq N_r(\mathcal{A})$. Note also that if $\mathcal{A}$ is monotone then so is $N_r(\mathcal{A})$. We can now state our generalized threshold DPT:

Theorem 2.5.2. *Fix a finite set B, and let $P \subseteq \{0,1\}^n \times B$ be a total relation for which $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) \leq 1 - \varepsilon$. Fixing any randomized algorithm $\mathcal{R}$ making queries to inputs $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$ and producing output in B^k, define the (random) set*

$$S[\mathbf{x}] \ := \ \{j \in [k] : (\mathbf{x}^j, \mathcal{R}_j(\mathbf{x})) \in P\}.$$

Suppose $\mathcal{R}$ is $\alpha\varepsilon Tk$-query-bounded for some $\alpha \in (0,1]$, and $\mathcal{A}$ is any monotone subset of $\mathcal{P}([k])$. Then:

1. $\Pr[S[\mathbf{x}] \in \mathcal{A}] \leq |B|^{\alpha\varepsilon k} \cdot \Pr[D \in \mathcal{A}]$, *where $D \subseteq [k]$ is generated by independently including each $j \in [k]$ in D with probability $1 - \varepsilon$.*

2. *Also, for D as above,* $\Pr[S[\mathbf{x}] \in \mathcal{A}] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})]$.

Proof. As in Theorem 2.0.1, we may assume $\varepsilon, T > 0$, $\mathrm{supp}(\mu) = \{0,1\}^n$. We have $\varepsilon \leq 1 - |B|^{-1} < 1$, since P is total and an algorithm may output a random element of B.

For $u \in \{0,1,*\}^n$ with $|u| \leq T$, let

$$W_P(u) \ := \ \mathrm{Suc}^{\mathrm{rel}}_{T-|u|,\mu^{(u)}}(P).$$

Then $W_P(u) \in [|B|^{-1}, 1]$. We have the following claim, whose proof follows that of Lemma 2.2.2:

Lemma 2.5.3. *1.* $W_P(*^n) \leq 1 - \varepsilon$.

2. For any $u \in \{0,1,*\}^n$ with $|u| < T$, and any $i \in [n]$, $\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W_P(u[x_i \leftarrow \mathbf{y}_i])] \leq W_P(u)$.

Let $\mathcal{R}$ be $\alpha\varepsilon Tk$-query-bounded; as in Theorem 2.0.1, we may assume $\mathcal{R}$ is deterministic, so call it $\mathcal{D}$ instead. Let $M := \lfloor \alpha\varepsilon Tk \rfloor$ as before, and recall the random strings u^j_t defined in Theorem 2.0.1.

Define random variables $\{X_{j,t}\}_{j \in [k], 0 \leq t \leq M}$, determined by an execution of $\mathcal{D}$ on inputs $(\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$, by letting $X_{j,t} := W_P(u^j_t)$ if $|u^j_t| \leq T$, otherwise $X_{j,t} := |B|^{-1}$. Next, the natural idea is to apply Lemma 2.5.1. First, however, we need to extend the sequences for one additional (non-query) step. That is, we will define random variables $X_{j,M+1}$ for each $j \in [k]$. We will use $\mathcal{X}$ to denote the collection of enlarged sequences.

Our definition of $X_{j,M+1}$ depends on whether $|u^j_M| \le T$, that is, on whether $\mathcal{D}$ made at most T queries to $\mathbf{x}^j$ on the current execution. If $|u^j_M| \le T$, let $X_{j,M+1} := \mathbf{1}_{[(\mathbf{x}^j, \mathcal{D}_j(\mathbf{x})) \in P]}$ be the indicator variable for the event that $\mathcal{D}$ solves P on the j-th input. If $|u^j_M| > T$, let $X_{j,M+1} := 1$ with probability $|B|^{-1}$, and let $X_{j,M+1} := 0$ with the remaining probability. We let each such "coin-flip" be independent of the others and of $(\mathbf{x}^1, \ldots, \mathbf{x}^k)$.

Define the collection $\mathcal{U} = \{U_0, \ldots, U_M\}$ by $U_t := (u^1_t, \ldots, u^k_t)$. We argue that the conditions of Lemma 2.5.1 are satisfied by $(\mathcal{X}, \mathcal{U})$, with $N := M + 1$. First, for $0 \le t' \le t \le M$, the stated conditions follow from Lemma 2.2.3 and part 2 of Lemma 2.5.3. Now consider the final, added step. Condition on any value of $U_M = (u^1_M, \ldots, u^k_M)$. Lemma 2.2.3 tells us that $\mathbf{x}^1, \ldots, \mathbf{x}^k$ are independent under this conditioning, and $\mathcal{D}$'s outputs are determined by U_M, so the variables $\{X_{j,M+1}\}$ are independent conditioned on U_M. If $|u^j_M| \le T$ then $\mathbb{E}[X_{j,M+1}|U_M] \le X_{j,M}$ by part 2 of Lemma 2.5.3. If $|u^j_M| > T$ then $\mathbb{E}[X_{j,M+1}] = |B|^{-1} = X_{j,M}$.

Thus the assumptions of Lemma 2.5.1 are satisfied, with $p_j = X_{j,0} \le 1 - \varepsilon$. We conclude that for any monotone $\mathcal{C} \subseteq \mathcal{P}([k])$,

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{C}] \;\le\; \Pr[D \in \mathcal{C}], \tag{2.4}$$

where each $j \in [k]$ is independently included in D with probability $1 - \varepsilon$.

To prove statement 1 of Theorem 2.5.2, let $\mathcal{C} := \mathcal{A}$. Note that $S[\mathbf{x}]$ and $u^1_M, \ldots, u^k_M$ are determined by $\mathbf{x}$, since $\mathcal{D}$ is deterministic. Condition on any value of $\mathbf{x}$ for which $S[\mathbf{x}] \in \mathcal{A}$. Under this conditioning, if $j \in [k]$ satisfies $|u^j_M| \le T$ and $j \in S[\mathbf{x}]$, then $X_{j,N} = 1$. On the other hand, if $|u^j_M| > T$, then $[X_{j,N} = 1]$ holds with probability $|B|^{-1}$, and these events are independent for each such j. By the query bound on $\mathcal{D}$, there are fewer than $\alpha\varepsilon k$ indices j in our conditioning for which $|u^j_M| > T$. Thus,

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in \mathcal{A} | S[\mathbf{x}] \in \mathcal{A}] \ \geq \ |B|^{-\alpha\varepsilon k},$$

which in combination with Eq. 2.4 implies

$$\Pr[S[\mathbf{x}] \in \mathcal{A}] \ \leq \ |B|^{\alpha\varepsilon k} \cdot \Pr[D \in \mathcal{A}],$$

as needed. To prove statement 2 of Theorem 2.5.2, let $\mathcal{C} := N_{\alpha\varepsilon k}(\mathcal{A})$ in Eq. 2.4: we find

$$\Pr[\{j \in [k] : X_{j,N} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})] \ \leq \ \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})].$$

Arguing as above, $S[\mathbf{x}] \setminus \{j \in [k] : X_{j,N} = 1\}$ is always a set of size less than $\alpha\varepsilon k$, so $[S[\mathbf{x}] \in \mathcal{A}]$ implies $[\{j \in [k] : X_{j,N} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})]$. Thus, we have $\Pr[S[\mathbf{x}] \in \mathcal{A}] \leq \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})]$. □

Part 1 of Theorem 2.5.2 is a proper generalization of Theorem 2.0.1. To see this, just set $\mathcal{A} := \{[k]\}$, $P := P_f$, and note that in this case, $\Pr[D \in \mathcal{A}] = (1 - \varepsilon)^k$. As another dividend, we obtain the following threshold DPT for relations, which specializes to an ordinary DPT for this setting (statement 3 in the Theorem below).

Theorem 2.5.4. *Let $P \subseteq \{0,1\}^n \times B$ be a total relation for which $\mathrm{Suc}^{\mathrm{rel}}_{T,\mu}(P) \leq 1-\varepsilon$. Fix any $\eta \in (0,1]$. For any randomized algorithm $\mathcal{R}$ making queries to inputs $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$, define the (random) set $S[\mathbf{x}]$ as in Theorem 2.5.2. Then if $\mathcal{R}$ is $\alpha\varepsilon Tk$-query-bounded for $\alpha \in (0,1]$, we have:*

1. $\Pr[|S[\mathbf{x}]| \geq \eta k] \leq |B|^{\alpha\varepsilon k} \cdot \Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq \eta k]$, *and also*

2. $\Pr[|S[\mathbf{x}]| \geq \eta k] \leq \Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (\eta - \alpha\varepsilon)k]$.

3. $\Pr[|S[\mathbf{x}]| = [k]]$ *is at most the minimum of* $|B|^{\alpha\varepsilon k}(1-\varepsilon)^k$ *and* $\Pr_{Y \sim B_{k,1-\varepsilon}}[Y \geq (1-\alpha\varepsilon)k]$. *If* $\alpha \leq 1/2$ *the second bound in the* min *is at most* $\left[1 - \varepsilon + 6\alpha\ln(1/\alpha)\varepsilon\right]^k$.

Proof. Apply parts 1 and 2 of Theorem 2.5.2, with the choice $\mathcal{A} := \{A \subseteq [k] : |A| \geq \eta k\}$. We have $\Pr[D \in \mathcal{A}] = \Pr[D_1 + \ldots + D_k \geq \eta k]$, where we define $D_j := \mathbf{1}_{[j \in D]}$.

These 0/1-valued variables are independent with bias $1 - \varepsilon$, which gives statement 1. Similarly, $\Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})] = \Pr[D_1 + \ldots + D_k \geq (\eta - \alpha\varepsilon)k]$, which gives statement 2. Statement 3 simply combines statements 1 and 2, under the setting $\eta = 1$. For the final bound in statement 3, we apply Lemma 2.1.2 with $\beta := \alpha$, $\delta := \varepsilon$. □

Theorem 2.0.4 in Section 2.0.5 follows from the special case of Theorem 2.5.4 in which $P := P_f$.

The success bound $|B|^{\alpha\varepsilon k}(1-\varepsilon)^k$ appearing above can also be derived by an easy modification of the proof of Theorem 2.0.1, in which the condition $X_{j,t} \geq 1/2$ we exploit becomes $X_{j,t} \geq |B|^{-1}$. When $|B|$ is large, however, the alternative bound provided in Theorem 2.5.4 will tend to give better results.

Note that part 2 of Theorem 2.5.4, in conjunction with Chernoff inequalities, gives success bounds which decay exponentially in k for any fixed $\alpha, \varepsilon, \eta$ for which $\eta > 1 - \varepsilon + \alpha\varepsilon$. Shaltiel's examples, described in Section 2.3, show that this cutoff is nearly tight: on those functions, the algorithm $\mathcal{D}$ described in Section 2.3 makes $2k + \alpha\varepsilon T k$ queries and (it is easily checked) typically solves about $(1 - \varepsilon + .5\alpha\varepsilon)k$ of the instances correctly.

Threshold DPTs for the worst-case setting can also be derived from Theorems 2.5.2 and 2.5.4, by the same reduction to the average-case setting used to prove Theorem 2.0.2.

2.5.3 Direct product theorems for learning tasks

Theorems 2.5.2 and 2.5.4 readily imply direct product theorems for the query complexity of certain learning tasks, as we explain next. Consider the scenario in which a randomized algorithm $\mathcal{R}$ is given query access to an unknown function $h : \{0,1\}^n \rightarrow \{0,1\}$ drawn from some distribution μ over a hypothesis class $\mathcal{H}$. That is, for any string x, $\mathcal{R}$ can query the value $h(x)$. The algorithm $\mathcal{R}$ attempts to output a hypothesis $\tilde{h}$ which is "close" to h. That is, we fix some symmetric relation $\mathbf{close} \subseteq \mathcal{H} \times \mathcal{H}$ (assume $\mathbf{close}(h, h)$ always holds), and we wish to find some $\tilde{h}$ such that $\mathbf{close}(h, \tilde{h})$ holds.

This task can be equivalently modeled as the relation problem associated with the total relation

$$P_{\mathcal{H}} \ := \ \{(h, h') : h, h' \in \mathcal{H} \wedge \mathbf{close}(h, h')\},$$

where h is given in truth-table form as a Boolean string, under the input distribution $h \sim \mu$. (We don't give a membership criterion for $P_{\mathcal{H}}$ when $h \notin \mathcal{H}$; this is unimportant since $\mathrm{supp}(\mu) \subseteq \mathcal{H}$.)

In the k-fold learning problem associated with $\mathcal{H}, \mu$, the algorithm has query access to each of k functions $(h_1, \ldots, h_k) \sim \mu^{\otimes k}$, and the goal is to output guesses $\tilde{h}_1, \ldots \tilde{h}_k$ such that $\mathbf{close}(h_j, \tilde{h}_j)$ holds for all (or at least "many") indices $j \in [k]$. This task is equivalent to the k-fold relation problem associated with $P_{\mathcal{H}}$, and Theorems 2.5.2 and 2.5.4 apply.

2.6 Proof of the XOR lemma

The proof of our XOR Lemma, Theorem 2.0.3 from Section 2.0.5, is modeled on the proof of our threshold DPTs, and reuses Lemma 2.5.1.

Proof of Theorem 2.0.3. As usual we first set up some preliminaries. For a deterministic algorithm $\mathcal{D}$ over n input bits define

$$W_{\oplus}(u) \ := \ 2 \cdot \mathrm{Suc}_{T-|u|, \mu^{(u)}}(f) - 1.$$

Lemma 2.6.1. *1.* $W_{\oplus}(*^n) \leq 1 - 2\varepsilon$.

2. For any $u \in \{0, 1, *\}^n$ *with* $|u| < T$*, and any* $i \in [n]$, $\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W_{\oplus}(u[x_i \leftarrow \mathbf{y}_i])] \leq W_{\oplus}(u)$.

Lemma 2.6.1 follows immediately from Lemma 2.2.2, since $W_{\oplus}(u) = 2W(u) - 1$.

Now we prove the Theorem. As in the proof of Theorem 2.0.1, we may assume $\varepsilon, T > 0$, $\mathrm{supp}(\mu) = \{0, 1\}^n$, and it is enough to prove the success bound for each deterministic $\alpha\varepsilon Tk$-query algorithm $\mathcal{D}$ attempting to solve $f^{\oplus k}(\mathbf{x}^1, \ldots, \mathbf{x}^k)$ on inputs $\mathbf{x}^1, \ldots, \mathbf{x}^k \sim \mu^{\otimes k}$. Recall the definitions of u_t^j (for $j \in [k], 0 \leq t \leq M := \lfloor \alpha\varepsilon Tk \rfloor$) from

Theorem 2.0.1. For a deterministic algorithm $\mathcal{D}$ define $\{X_{j,t}\}_{j\in[k],0\le t\le M}$ as follows: if $|u^j_t| \le T$, set $X_{j,t} := W_\oplus(u^t_j)$; otherwise, set $X_{j,t} := 0$.

We will extend the random sequences $\{X_{j,t}\}$ for one additional (non-query) step, and will let $\mathcal{X}$ denote our enlarged collection. To set up our extension, we first define random variables b_j, r_j, a_j for $j \in [k]$, determined by u^j_M, as follows. Let $b_j \in \{0,1\}$ be defined as the likeliest value of $f(\mathbf{y})$, where $\mathbf{y} \sim \mu^{(u^j_M)}$ (break ties arbitrarily). Let $r_j := \Pr[f(\mathbf{y}) = b_j] \in [1/2, 1]$, where again $\mathbf{y} \sim \mu^{(u^j_M)}$. Let $a_j := 2r_j - 1 \in [0,1]$.

If $|u^j_M| > T$, set $X_{j,M+1} := 0$. If instead $|u^j_M| \le T$, our random process "inspects" the actual value of the bit $f(\mathbf{x}^j)$ to help determine $X_{j,M+1}$. If $f(\mathbf{x}^j) \ne b_j$, let $X_{j,M+1} := 0$. If $f(\mathbf{x}^j) = b_j$, let $X_{j,M+1} := 1$ with probability a_j/r_j, and $X_{j,M+1} := 0$ with the remaining probability, where this random decision is independent of all others. Thus in this case,

$$\mathbb{E}[X_{j,M+1}|u^1_M, \ldots, u^k_M] \;=\; r_j \cdot (a_j/r_j) \;=\; a_j \;\le\; X_{j,M},$$

where the last inequality holds by the definition of $W_\oplus(u^j_M)$ since $|u^j_M| \le T$.

Let $\mathcal{U} = (U_0, \ldots, U_M)$, where $U_t := (u^1_t, \ldots, u^k_t)$. By an argument analogous to that in the proof of Theorem 2.5.2, we verify that $(\mathcal{X},\mathcal{U})$ obey the assumptions of Lemma 2.5.1, this time with $p_j := 1 - 2\varepsilon$. Applying Lemma 2.5.1 to $\mathcal{A} := \{A \subseteq [k] : |A| > (1 - \alpha\varepsilon)k\}$, we find

$$\Pr[|\{j : X_{j,M+1} = 1\}| > (1 - \alpha\varepsilon)k] \;\le\; \Pr[D \in \mathcal{A}], \tag{2.5}$$

where each $j \in [k]$ is independently included in D with probability $(1-2\varepsilon)$. We have $\Pr[D \in \mathcal{A}] = \Pr_{Y\sim B_{k,1-2\varepsilon}}[Y > (1 - \alpha\varepsilon)k]$.

We analyze events F of form $F := [U_M = (u^1_M, \ldots, u^k_M), X_{1,M+1} = z_1, \ldots, X_{k,M+1} = z_k]$. Note that conditioning on F does *not* condition on the particular values $f(\mathbf{x}^j)$ which helped determine the values z_j. Focus attention on any such event F for which $|\{j : X_{j,M+1} = 1\}| \le (1-\alpha\varepsilon)k$. Since $\mathcal{D}$ makes at most $\alpha\varepsilon Tk$ queries, there are fewer than $\alpha\varepsilon k$ indices j for which $|u^j_M| > T$. In particular, there exists a $j^\star \in [k]$ for which $|u^{j^\star}_M| \le T$ *and* $X_{j^\star,M+1} < 1$ (so, by our definitions, $X_{j^\star,M+1} = 0$).

Now let the event F' be defined just like F, except that F' makes no conditioning on $X_{j^\star,M+1}$ (so, $F = F' \wedge [X_{j^\star,M+1} = 0]$). Then,

$$\begin{aligned}
\Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star}|F] &= \Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star}|F' \wedge X_{j^\star,M+1} = 0] \\
&= \frac{\Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star} \wedge X_{j^\star,M+1} = 0|F']}{\Pr[X_{j^\star,M+1} = 0|F']} \\
&= \frac{\Pr[f(\mathbf{x}^{j^\star}) = b_{j^\star}|F'] \cdot \Pr[X_{j^\star,M+1} = 0|F', f(\mathbf{x}^{j^\star}) = b_{j^\star}]}{\sum_{b\in\{0,1\}} \Pr[f(\mathbf{x}^{j^\star}) = b|F'] \cdot \Pr[X_{j^\star,M+1} = 0|F', f(\mathbf{x}^{j^\star}) = b]} \\
&= \frac{r_{j^\star}(1 - a_{j^\star}/r_{j^\star})}{r_{j^\star}(1 - a_{j^\star}/r_{j^\star}) + (1 - r_{j^\star}) \cdot 1}
\end{aligned}$$

(using the fact that $\mathbf{x}^1, \ldots, \mathbf{x}^k$ are independent conditioned on U_M, by Lemma 2.2.3, and the additional fact that $\{X_{j,M+1}\}_{j\in[k]}$ are independent conditioned on U_M)

$$= \frac{r_{j^\star} - a_{j^\star}}{1 - a_{j^\star}} = \frac{\frac{1}{2}(1 + a_{j^\star}) - a_{j^\star}}{1 - a_{j^\star}} = 1/2.$$

Thus, $f(\mathbf{x}^{j^\star})$ is an unbiased random bit conditioned on F. Consequently, $f^{\oplus k}(\mathbf{x}^1, \ldots, \mathbf{x}^k) = f(\mathbf{x}^{j^\star}) \oplus f^{\oplus k-1}(\mathbf{x}^1, \ldots, \mathbf{x}^{j^\star-1}, \mathbf{x}^{j^\star+1}, \ldots, \mathbf{x}^k)$ is an unbiased random bit conditioned on F. Thus under this conditioning, $\mathcal{D}$'s output bit equals the k-fold XOR with probability exactly 1/2. Now F was an arbitrary outcome of $U_M, X_{1,M+1}, \ldots, X_{k,M+1}$ for which $|\{j : X_{j,M+1} = 1\}| \le (1 - \alpha\varepsilon)k$. It follows that

$$\begin{aligned}
\Pr_{\mathbf{x}\sim\mu^{\otimes k}}[\mathcal{D}(\mathbf{x}) = f^{\oplus k}(\mathbf{x})] &\le \Pr[|\{j : X_{j,M+1} = 1\}| > (1 - \alpha\varepsilon)k] + \\
&\qquad \frac{1}{2}\Pr[|\{j : X_{j,M+1} = 1\}| \le (1 - \alpha\varepsilon)k] \\
&= \frac{1}{2}(1 + \Pr[|\{j : X_{j,M+1} = 1\}| > (1 - \alpha\varepsilon)k]) \\
&\le \frac{1}{2}\left(1 + \Pr_{Y\sim B_{k,1-2\varepsilon}}[Y > (1 - \alpha\varepsilon)k]\right),
\end{aligned}$$

using Eq. 2.5.

Finally, to get the concrete bound claimed in statement of Theorem 2.0.3, first suppose $\varepsilon = 1/2$; in this case the bound follows easily since $Y = 0$ with certainty. If $\varepsilon < 1/2$, note that $(1 - \alpha\varepsilon)k = (1 - (\alpha/2)(2\varepsilon))$, and apply Lemma 2.1.2 with

$\delta := 2\varepsilon < 1$ and $\beta := \alpha/2 \leq 1/2$. $\square$

2.7 Direct product theorems for search problems and errorless heuristics

We define a fairly general notion of search problems in the query model for which a direct product theorem can be proved. We will also obtain a DPT for *errorless heuristics*, defined in Section 2.7.2.

2.7.1 Search problems

We need some preliminary definitions. Given $u, v \in \{0, 1, *\}^n$, say that u and v *agree* if $u_i \in \{0, 1\}$ implies $v_i \in \{*, u_i\}$. Note that this definition is symmetric in u and v. If u, v agree, define their *overlay* $u \circ v \in \{0, 1, *\}^n$ by $(u \circ v)_i := b \in \{0, 1\}$ if either $u_i = b$ or $v_i = b$, otherwise $(u \circ v)_i := *$. Say that u *extends* v if $v_i \in \{0, 1\}$ implies $u_i = v_i$.

Say we are given a distribution μ on $\{0, 1\}^n$, and a (possibly randomized) query algorithm $\mathcal{R}$; if $\mathcal{R}$ runs on an input distributed according μ, we denote by $U_{\mathcal{R},\mu} \in \{0, 1, *\}^n$ the random string describing the input bits seen by $\mathcal{R}$.

A *search problem* is defined by a subset $V \subseteq \{0, 1, *\}^n$. We say that $\mathcal{R}$ *ε-solves* the search problem V with respect to an input distribution μ over $\{0, 1\}^n$ if, with probability $\geq 1 - \varepsilon$, $U_{\mathcal{R},\mu}$ extends some $v \in V$. (We allow the possibility that some $x \in \text{supp}(\mu)$ do not extend *any* $v \in V$.) Define $\text{Suc}_{T,\mu}(V) := 1 - \varepsilon$, where ε is the minimal value such that some T-query randomized algorithm ε-solves search problem V on inputs from μ.

Define the k-fold search problem $V^{\otimes k} := \{(v^1, \ldots, v^k) : v^j \in V, \forall j \in [k]\} \subseteq \{0, 1, *\}^{kn}$. Thus to solve $V^{\otimes k}$, an algorithm must solve each of the k constituent search problems. We generalize this notion in order to state a threshold DPT, which will imply our ordinary DPT. For a monotone subset $\mathcal{A} \subseteq \mathcal{P}([k])$, define

$$V^{k,\mathcal{A}} \;:=\; \{(v^1, \ldots, v^k) : \{j \in [k] : v^j \in V\} \in \mathcal{A}\}.$$

Thus to solve $V^{k,\mathcal{A}}$, an algorithm must solve "sufficiently many" of the k search problems, as specified by $\mathcal{A}$.

Recall the notation $N_r(\cdot)$ from Section 2.5. Our generalized threshold DPT for search problems is as follows:

Theorem 2.7.1. *Suppose the search problem V satisfies $\mathrm{Suc}_{T,\mu}(V) \leq 1 - \varepsilon$. Then for any $\alpha \in (0, 1]$ and any monotone $\mathcal{A} \subseteq \mathcal{P}([k])$,*

$$\mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(V^{k,\mathcal{A}}) \;\leq\; \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})],$$

where each $j \in [k]$ is independently included in D with probability $1 - \varepsilon$.

Proof. In the search setting, ε can potentially be any value in $[0, 1]$. The boundary cases are trivial, so assume $0 < \varepsilon < 1$. As usual, we can assume that $T > 0$ and $\mathrm{supp}(\mu) = \{0, 1\}^n$, and it is enough to bound the success probability of any deterministic $\alpha\varepsilon Tk$-query algorithm.

Following Theorem 2.0.1, we first develop some concepts related to a computation on a single input to the search problem V. For each $u \in \{0, 1, *\}^n$ for which $|u| \leq T$, let $\mathrm{Val}_V(u) := 1$ if u extends some $v \in V$, otherwise $\mathrm{Val}_V(u) := 0$. For a deterministic query algorithm $\mathcal{D}$ let $W_V(u, \mathcal{D}) := \mathbb{E}[\mathrm{Val}(u \circ U_{\mathcal{D},\mu^{(u)}})]$. (Note that u and $U_{\mathcal{D},\mu^{(u)}}$ always agree.)

If $|u| \leq T$, let $W_V(u) := \max_{\mathcal{D}}(W_V(u, \mathcal{D}))$, where the maximum ranges over all deterministic algorithms making at most $T - |u|$ queries. In other words, $W_V(u)$ is the maximum success probability of any $(T - |u|)$-query algorithm in solving V on an input $\mathbf{y} \sim \mu^{(u)}$, where we reveal the bits described by u "for free" to the algorithm. Then we have:

Lemma 2.7.2. *1.* $W_V(*^n) \leq 1 - \varepsilon$.

2. For any $u \in \{0,1,*\}^n$ with $|u| < T$, and any $i \in [n]$, $\mathbb{E}_{\mathbf{y}\sim\mu^{(u)}}[W_V(u[x_i \leftarrow \mathbf{y}_i])] \leq W_V(u)$.

We omit the proof, which is essentially the same as that of Lemma 2.2.2.

Let $\mathcal{D}$ be any deterministic algorithm making at most $M := \lfloor \alpha\varepsilon T k\rfloor$ queries and attempting to compute $V^{k,\mathcal{A}}$ on inputs drawn as $(\mathbf{x}^1,\ldots,\mathbf{x}^k) \sim \mu^{\otimes k}$. For $0 \leq t \leq M$, and for $j \in [k]$, let u_t^j be defined as in the previous proofs. Let $\mathcal{X} = \{X_{j,t}\}_{j\in[k],0\leq t\leq M}$, where $X_{j,t} := W_V(u_t^j)$ if $|u_t^j| \leq T$, otherwise $X_{j,t} := 0$.

Unlike in Theorem 2.5.2, we have no need to add any additional steps to our random sequences. For $0 \leq t < M$, we let $U_t := (u_t^1,\ldots,u_t^k)$ just as before. Setting $N := M$ and reasoning as in Theorem 2.5.2, we verify that the assumptions of 2.5.1 are satisfied, with $p_j = X_{j,0} \leq 1-\varepsilon$ (Lemma 2.7.2, part 1).

Applying Lemma 2.5.1 to the monotone set $N_{\alpha\varepsilon k}(\mathcal{A})$, we conclude that

$$\Pr[\{j \in [k] : X_{j,M} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})] \;\leq\; \Pr[D \in N_{\alpha\varepsilon k}(\mathcal{A})], \tag{2.6}$$

where each $j \in [k]$ is independently included in D with probability $1-\varepsilon$.

Now condition on any execution of $\mathcal{D}$, and consider any $j \in [k]$ such that $X_{j,M} < 1$. By our definitions, at least one of two possibilities holds: either $|u_M^j| > T$ (there are fewer than $\alpha\varepsilon k$ such indices j), or u_M^j does not extend any $v \in V$. Thus if $\mathcal{D}$ solves the search problem $V^{k,\mathcal{A}}$ on the present execution, we have $\{j \in [k] : X_{j,M} = 1\} \in N_{\alpha\varepsilon k}(\mathcal{A})$. Combining this with Eq. 2.6 yields the Theorem. □

From Theorem 2.7.1, we will directly obtain a standard threshold DPT and an ordinary DPT for search problems. First, given a search problem $V \subseteq \{0,1,*\}^n$ and a real number $s \in [0,k]$, define $\mathcal{C}[\geq s] := \{A \subseteq [k] : |A| \geq s\}$.

Theorem 2.7.3. *Suppose* $\mathrm{Suc}_{T,\mu}(V) \leq 1-\varepsilon$. *Then for any* $\alpha \in (0,1]$ *and any* $\eta \in (0,1]$,

$$\mathrm{Suc}_{\alpha\varepsilon Tk,\mu^{\otimes k}}(V^{k,\mathcal{C}[\geq \eta k]}) \;\leq\; \Pr_{Y\sim B_{k,1-\varepsilon}}[Y > (\eta-\alpha\varepsilon)k]\,.$$

Proof. Apply Theorem 2.7.1 with $\mathcal{C} := \mathcal{C}[\geq \eta k]$, and note that $D \in N_{\alpha\varepsilon k}(\mathcal{C}[\geq \eta k])$ iff $|D| > \eta k - \alpha\varepsilon k$, which is equivalent to $[D_1 + \ldots + D_k > (\eta-\alpha\varepsilon)k]$, where $D_j :=$

$\mathbf{1}_{[j \in D]}$. These indicator variables are independent with expectation $1 - \varepsilon$. □

Theorem 2.7.4. *Suppose* $\mathrm{Suc}_{T,\mu}(V) \leq 1 - \varepsilon$. *Then for any* $\alpha \in (0, 1]$,

$$\mathrm{Suc}_{\alpha\varepsilon Tk, \mu^{\otimes k}}(V^{\otimes k}) \ \leq \ \Pr_{Y \sim B_{k,1-\varepsilon}} [Y > (1 - \alpha\varepsilon)k] \,.$$

Proof. Note that $V^{\otimes k} = V^{k, \mathcal{C}[\geq k]}$, so the result follows from Theorem 2.7.3 with $\eta := 1$. □

2.7.2 Errorless heuristics

An *errorless heuristic* for a (not necessarily Boolean) function $f : \{0,1\}^n \to B$ is a randomized query algorithm $\mathcal{R}$ outputting values in $B \cup \{?\}$ such that for all x, $\mathcal{R}(x) \in \{f(x), ?\}$ with probability 1. We say that an errorless heuristic $\mathcal{R}$ *ε-solves f with zero error* with respect to input distribution μ if $\Pr_{x \sim \mu}[\mathcal{R}(x) = f(x)] \geq 1 - \varepsilon$. Let $\mathrm{Suc}^{\text{0-err}}_{T,\mu}(f) := 1 - \varepsilon$, where ε is the minimal value such that some T-query errorless heuristic ε-solves f with zero error with respect to μ. Note that $\mathrm{Suc}^{\text{0-err}}_{T,\mu}(f)$ is exactly $\mathrm{Suc}_{T,\mu}(V_f)$, where the search problem V_f is defined as

$$V_f \ := \ \{u \in \{0, 1, *\}^n : u \text{ forces the value of } f\}.$$

Also, note that $V_{f^{\otimes k}} = V_f^{\otimes k}$. Thus the following result is immediately implied by Theorem 2.7.4:

Theorem 2.7.5. *Suppose* $\mathrm{Suc}^{\text{0-err}}_{T,\mu}(f) \leq 1 - \varepsilon$. *Then for* $\alpha \in (0, 1]$,

$$\mathrm{Suc}^{\text{0-err}}_{\alpha\varepsilon Tk, \mu^{\otimes k}}(f^{\otimes k}) \ \leq \ \Pr_{Y \sim B_{k,1-\varepsilon}} [Y > (1 - \alpha\varepsilon)k] \,.$$

Let us revisit the XOR problem in the current setting. It is easy to see that an errorless heuristic to compute the k-fold XOR $f^{\oplus k}$, on inputs drawn from a product distribution, cannot produce any output other than " ? " unless its queries allow it to determine the value of $f^{\otimes k}$. Thus Theorem 2.7.5 also implies an XOR lemma with the same success bound for errorless heuristics.

Next we prove a worst-case analogue of Theorem 2.7.5. Define $R_0(f)$, the *zero-error randomized query complexity of* f, as the minimum T for which some algorithm $\mathcal{R}$ outputs $f(x)$ with probability 1 for each x, and for which the *expected* number of queries made by $\mathcal{R}$ to any input is at most T. The following is another variant of Yao's minimax principle [Yao77]; we include a proof for completeness.

Lemma 2.7.6. *Let $\eta \in (0, 1]$. There exists a distribution μ_η over inputs to f, such that* $\mathrm{Suc}^{\text{0-err}}_{\eta R_0(f),\mu_\eta}(f) \leq \eta$.

Proof. Consider the following 2-player game: player 1 chooses a (possibly randomized) errorless heuristic $\mathcal{R}$ for f which makes at most $\eta R_0(f)$ queries, and player 2 chooses (simultaneously) an input x to f. Player 1 wins if $\mathcal{R}(x) = f(x)$. We claim there exists a randomized strategy for player 2, that is, a distribution $\mu =: \mu_\eta$ over inputs to x, that beats any strategy of player 1 with probability at least $1 - \eta$. This will prove the Lemma.

To prove the claim, suppose for contradiction's sake that no such strategy for player 2 exists. Then, by the minimax theorem, there exists a randomized strategy for player 1 which wins with probability greater than η against all choices of x. This strategy is itself a randomized algorithm making at most $\eta R_0(f)$ queries; let us call this algorithm $\mathcal{R}$. Consider the algorithm $\mathcal{R}'$ for f that on input x, repeatedly applies $\mathcal{R}$ to x until $\mathcal{R}$ produces an output, which $\mathcal{R}'$ then outputs. We have $\mathcal{R}'(x) = f(x)$ on every input. Also, the expected number of queries of $\mathcal{R}'$ on any input is strictly less than

$$\begin{aligned}
\sum_{m\geq 1}(1-\eta)^{m-1}\eta\,(m\cdot\eta R_0(f)) &= \left(\sum_{m\geq 1}(1-\eta)^{m-1}m\right)\cdot\eta^2 R_0(f)\\
&= \frac{1}{\eta^2}\cdot\eta^2 R_0(f)\\
&= R_0(f),
\end{aligned}$$

contradicting the definition of $R_0(f)$. □

Theorem 2.7.7. *For any (not necessarily Boolean) function f, and $\alpha \in (0, 1/2]$, any* *errorless heuristic for $f^{\otimes k}$ using at most $\alpha^2 R_0(f)k/4$ queries has worst-case success probability less than $(7\alpha\ln(1/\alpha))^k$.*

Proof. Set $\gamma := \alpha/2$. Let μ_γ be the distribution given by Lemma 2.7.6, so that $\mathrm{Suc}^{\text{0-err}}_{\gamma R_0(f),\mu_\gamma}(f) \le \gamma$. By Theorem 2.7.5 applied to α, with $T := \gamma R_0(f)$ and $\varepsilon := 1-\gamma$,

$$\mathrm{Suc}^{\text{0-err}}_{\alpha(1-\gamma)\gamma R_0(f)k,\mu_\delta^{\otimes k}}(f^{\otimes k}) \;\le\; \Pr_{Y\sim B_{k,\gamma}}\left[Y > (1-\alpha(1-\gamma))k\right].$$

We have $\alpha^2 R_0(f)k/4 \le \alpha(1-\gamma)\gamma R_0(f)k$ (using $\gamma \le 1/2$), so that

$$\begin{aligned}\mathrm{Suc}^{\text{0-err}}_{\alpha^2 R_0(f)k/4,\mu_\gamma^{\otimes k}}(f^{\otimes k}) \;&\le\; \Pr_{Y\sim B_{k,\gamma}}\left[Y > (1-\alpha(1-\gamma))k\right] \\ &<\; \left[1-(1-\gamma)+6\alpha\ln(1/\alpha)(1-\gamma))\right]^k\end{aligned}$$

(applying Lemma 2.1.2, with $\beta := \alpha \le 1/2$ and $\delta := (1-\gamma)$)

$$\begin{aligned}&<\; (\alpha/2 + 6\alpha\ln(1/\alpha))^k \\ &<\; (7\alpha\ln(1/\alpha))^k\,.\end{aligned}$$

□

2.8 A direct product theorem for decision tree size

We measure the *size* of a decision tree $\mathcal{D}$, denoted size($\mathcal{D}$), as the number of leaf (output) vertices. Note that this is at least 1/2 the total number of vertices. Define $\mathrm{Suc}^{\text{size}}_{T,\mu}(f)$ as the maximum success probability of any size-T decision tree attempting to compute f on an input drawn from distribution μ. We have the following DPT for size-bounded query algorithms:

Theorem 2.8.1. *Let f be a Boolean function. Suppose $\mathrm{Suc}^{\text{size}}_{T,\mu}(f) \le 1-\varepsilon$. Then for $0 < \alpha \le 1$, $\mathrm{Suc}^{\text{size}}_{T^{\alpha\varepsilon k},\mu^{\otimes k}}(f^{\otimes k}) \le 2^{\alpha\varepsilon k}(1-\varepsilon)^k$.*

Note how the size bound grows exponentially, rather than linearly, in k in the above statement. It is natural to expect such a statement, since the k-fold application of a size-T decision tree is described by a size-T^k decision tree. Also note that, by convexity, Theorem 2.8.1 also bounds the success probability of any "randomized size-$T^{\alpha\varepsilon k}$ algorithm" $\mathcal{R}$, i.e., of any probability distribution over size-$T^{\alpha\varepsilon k}$ decision trees.

Proof. The proof follows that of Theorem 2.0.1, except that we need a new way to quantify the resources used by each of the k inputs. First we develop some definitions pertaining to a single input to f. Given $u \in \{0,1,*\}^n$ and a real number $Z \in [1,T]$, let

$$W_{\text{size}}(u,Z) \ := \ \text{Suc}^{\text{size}}_{Z,\mu^{(u)}}(f).$$

Lemma 2.8.2. *1.* $W_{\text{size}}(*^n, T) \le 1-\varepsilon$.

2. Take any real numbers $S^{(0)}, S^{(1)} \ge 1$ *and let* $S := S^{(0)} + S^{(1)}$. *Then for any* $u \in \{0,1,*\}^n$ *and any* $i \in [n]$,

$$\mathbb{E}_{\mathbf{y}\sim\mu^{(u)}}[W_{\text{size}}(u[x_i \leftarrow \mathbf{y}_i], S^{(\mathbf{y}_i)})] \ \le \ W_{\text{size}}(u,S).$$

The proof is very similar to that of Lemma 2.2.2, and is omitted.

Now let $\mathcal{D}$ be any deterministic algorithm of size at most $T^{\alpha\varepsilon k}$ attempting to compute $f^{\otimes k}$ on input strings $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$. Let $M := \lfloor T^{\alpha\varepsilon k} \rfloor$; $\mathcal{D}$ always makes at most M queries.

As in previous proofs, for $j \in [k]$ and $0 \le t \le M$, let $u^j_t \in \{0,1,*\}^n$ describe the outcomes of all queries made to $\mathbf{x}^j$ after $\mathcal{D}$ has taken t steps (here a "step" consists of a query, unless $\mathcal{D}$ has halted, in which case a step has no effect).

Let S_t be defined as the size (number of leaf vertices) of the subtree of $\mathcal{D}$ reached after t steps have been taken. Thus we have $S_0 \le T^{\alpha\varepsilon k}$, and $S_t = 1$ iff $\mathcal{D}$ has halted after at most t queries. For each $j \in [k]$, we define a sequence $Z_{j,0}, \ldots, Z_{j,M}$, as follows. Let $Z_{j,0} := T$. For $0 \le t < M$, if $\mathcal{D}$ has halted after t steps, let $Z_{j,t+1} := Z_{j,t}$. Otherwise, if the $(t+1)$-st query made by $\mathcal{D}$ is *not* to $\mathbf{x}^j$, we again let $Z_{j,t+1} := Z_{j,t}$.

If the $(t+1)$-st query is to $\mathbf{x}^j$, let

$$Z_{j,t+1} \;:=\; \frac{S_{t+1}}{S_t} \cdot Z_{j,t}.$$

Let $X_{j,t} := W_{\text{size}}(u_j^t, Z_{j,t})$ if $Z_{j,t} \geq 1$; otherwise let $X_{j,t} := 1/2$. Let $P_t := \prod_{j \in [k]} X_{j,t}$. Arguing as in Theorem 2.0.1, for each $0 \leq t < M$, $\mathbb{E}[P_{t+1}] \leq \mathbb{E}[P_t]$. It follows that $\mathbb{E}[P_M] \leq \mathbb{E}[P_0] = W_{\text{size}}(*^n, T)^k \leq (1-\varepsilon)^k$.

Condition on any complete execution of $\mathcal{D}$, as described by $u_M^1, \ldots, u_M^k$. Notice that if $Z_{j,M} \geq 1$, then (by the definitions) $X_{j,M}$ is an upper bound on the conditional success probability of guessing $f(\mathbf{x}^j)$ correctly. Also, $X_{j,t} \geq 1/2$ for all j, t, and all inputs are independent after our conditioning. Thus the conditional success probability of computing $f^{\otimes k}(\mathbf{x})$ is at most $2^{|B|} \cdot P_M$, where we define the (random) set $B := \{j \in [k] : Z_{j,M} < 1\}$.

Observe that $S_M = 1$, since the algorithm halts after at most M steps. Then,

$$\begin{aligned} 1 \;=\; S_M \;&=\; \left(\frac{S_1}{S_0}\right) \cdot \ldots \cdot \left(\frac{S_M}{S_{M-1}}\right) \cdot S_0 \\ &\leq\; \left(\frac{\prod_{j \in [k]} Z_{j,M}}{T^k}\right) \cdot T^{\alpha\varepsilon k} \\ &\leq\; T^{-|B|} \cdot T^{\alpha\varepsilon k}. \end{aligned}$$

Thus, $|B| \leq \alpha\varepsilon k$ always. So the *overall* success probability is at most $\mathbb{E}[2^{|B|} P_M] \leq 2^{\alpha\varepsilon k}\mathbb{E}[P_M] \leq (2^{\alpha\varepsilon}(1-\varepsilon))^k$. □

One can also prove variants of our XOR lemma and other results in which we impose bounds on decision tree size rather than number of queries. We omit the details.

2.9 DPTs for dynamic interaction

So far, all of the computational tasks we have studied have involved algorithms querying a collection of fixed input strings. However, in many situations in computer science it is natural to consider more general problems of *interaction* with dynamic, stateful entities. An algorithm can still "query" these entities, but these actions may influence the outcomes of future queries. In this section we describe how our proof methods can yield DPTs for these more general problems. The methods involved are essentially the same as in previous sections, and the theorem we give is just one example of the kind of DPT we can prove for dynamic interaction, so we will only sketch the proofs here, indicating the novel elements.

We will propose a self-contained model of dynamic interaction. We make no claims of conceptual novelty for this model, however. Dynamic interaction has been an important concept for cryptography; in this context, Maurer [Mau02] proposed a model of *random systems* that generalizes our model. All of our work in this section could in principle be carried out in the random systems framework; we choose to use a different model that is somewhat simpler and adequate to our needs, and that preserves a clear resemblance to our work in previous sections.

Much of the work in the random systems framework studies various kinds of *composition* of random systems; this work aims to understand how cryptographic primitives can be combined into more complex protocols. In this vein, Maurer, Pietrzak, and Renner [MPR07] proved a result (see their Lemma 6) that can be informally described as follows: if an agent is playing games with two or more independent, non-communicating entities, then the maximum joint-success probability is achieved by following independent strategies on the different games. This result establishes an "ideal" direct product property for interaction tasks with k independent entities, in which the number of queries to each entity is fixed in advance. By contrast, our focus will be on proving DPTs for query algorithms that can adaptively reallocate queries between the k entities.

Now we formally define the type of entity with which our query algorithms interact. Define an *interactive automaton (IA)* as a 5-tuple

$$\mathcal{M} \;=\; (\mathbf{seeds}, \mathbf{states}, \mathbf{queries}, R, \Delta), \text{ where:}$$

- **seeds**, **states**, **queries** are each finite sets, and **states** contains a distinguished *start state* s_0;

- R : **seeds** $\times$ **states** $\times$ **queries** $\rightarrow \{0,1\}$ is a *response mapping*;

- Δ : **seeds** $\times$ **states** $\times$ **queries** $\rightarrow$ **states** is a *transition mapping.*

These automata are deterministic, but we can incorporate randomness by providing random bits as part of **seeds**.

We consider the scenario in which $\mathcal{M}$ is initialized to some seed $z \in$ **seeds** according to a distribution μ, along with the start-state s_0. The automaton retains the value z throughout an interaction with a query algorithm $\mathcal{R}$ (which does not know the value z), but changes its state-value. If $\mathcal{R}$ selects the query $q \in Q$ while $\mathcal{M}$ has internal state $(z, s) \in$ **seeds** $\times$ **states**, then $\mathcal{M}$ returns the value $R(z, s, q)$ to $\mathcal{R}$ and transitions to the state $(z, \Delta(z, s, q))$.[4]

There are several kinds of tasks one can associate with an IA. One such task for the query algorithm $\mathcal{R}$ is to try to output a value $b \in B$ that satisfies some predicate $P(z, b)$, where z is the seed to $\mathcal{M}$ and $P \subseteq$ **seeds** $\times B$ is a total relation over **seeds** and a finite set B. This, of course, is a generalization of the relation problems we studied in Section 2.5, and it is natural to study the k-fold setting, in which $\mathcal{R}$ interacts with k IAs, querying one of them at each step. We assume that each IA only updates its state or sends a response to $\mathcal{R}$ when it is queried. In particular, the IAs do not communicate with each other.

We can transform the IA interaction scenario into an equivalent one which highlights the similarity with the standard query model, and makes it easy to apply our previous work to obtain a DPT. For simplicity assume $|\,\textbf{seeds}\,| = 2^m$. Given an IA $\mathcal{M}$ and an integer $N > 0$, for each $z \in$ **seeds** we define a string $\xi(z) \in \{0,1\}^{m+(|\,\textbf{queries}\,|+1)^N}$. There are two types of entries in this string. First there are m "ID" entries, which simply contain a binary encoding of z. Next there are

[4]We can now sketch the modeling differences between our work and [Mau02]. Maurer's "random systems" are modeled as inherently randomized; they may or may not be finite-state machines; and they are specified "behaviorally" by their conditional distributions over query responses, conditioned on all possible conversation transcripts.

$(|\,\mathbf{queries}\,| + 1)^N$ "response" entries, with each such entry indexed by an N-tuple $\overline{q} = (q_1, \ldots, q_N) \in (\mathbf{queries} \cup \{*\})^N$. We are only interested in response-entries of form $\overline{q} = (q_1, \ldots, q_r, *, *, \ldots, *)$, where $q_1, \ldots, q_r \in \mathbf{queries}$. For such an entry we define $\xi(z)_{\overline{q}} \in \{0, 1\}$ as the result of the following experiment: initialize $\mathcal{M}$ to state (z, s_0), and perform the interaction in which a query algorithm asks queries $q_1, \ldots, q_r$ in that order. Let $\xi(z)_{\overline{q}}$ be the final, r-th response made by $\mathcal{M}$.

Define a total relation $P_\xi \subseteq \{0,1\}^{m+(|\,\mathbf{queries}\,|+1)^N} \times B$ by

$$P_\xi \ := \ \{(\xi(z), b) : z \in \mathbf{seeds} \wedge P(z, b)\}.$$

Also, given a distribution μ over $\mathbf{seeds}$, define $\mu_\xi \sim \xi(z)$, where $z \sim \mu$. In this way we map an IA interaction task onto a relation problem of the type studied in Section 2.5, with a corresponding map from initialization distributions to input distributions.

A standard query algorithm $\mathcal{R}$ (as studied in all previous sections) can faithfully simulate an interaction with $\mathcal{M}$ initialized to an unknown $z \in \mathbf{seeds}$, if given query access to $\xi(z)$. This works in the natural way: if its simulated queries up to the r-th step are $q_1, \ldots, q_r$, then for its r-th query to $\xi(z)$, $\mathcal{R}$ looks at the entry $(q_1, \ldots, q_r, *, *, \ldots, *)$ to learn $\mathcal{M}$'s r-th response. Call an algorithm "interaction-faithful" if its sequence of queries to any input string always obeys this format.

Of course, not all algorithms are interaction-faithful. For example, an unfaithful algorithm could simply look at the ID-entries to learn z. Thus the relation problem (P_ξ, μ_ξ) can be much easier than the IA interaction problem defined by $(\mathcal{M}, P, \mu)$. However, if we restrict attention to the class of interaction-faithful algorithms $\mathcal{R}$, then it is not hard to see that there is an *exact* correspondence between the "difficulty" of the two problems, at least for interactions lasting at most N steps. That is, for $T \le N$, there is a T-query IA-interaction algorithm for $(\mathcal{M}, P, \mu)$ with success probability p, if and only if there is a T-query interaction-faithful standard algorithm for (P_ξ, μ_ξ) with success probability p.

The good news is that we can prove a DPT for interaction-faithful query algorithms in almost exactly the same way as for unrestricted query algorithms. In fact, it's most natural to prove a DPT for a more general notion of faithfulness, which we define next. Say we are given $n > 0$ and a map $\tau : \{0,1,*\}^n \to \{0,1\}^n$, called a *query-restriction map.* Say that a (standard) query algorithm $\mathcal{R}$ on n input bits is *τ-faithful* if for every execution of $\mathcal{R}$ on any input, whenever the input bits seen by $\mathcal{R}$ seen so far are given by $u \in \{0,1,*\}^n$, then $\mathcal{R}$ either halts, or chooses a next input bit x_i to query whose index satisfies $\tau(u)_i = 1$. In other words, a restriction map τ restricts the possible next queries which can be made by a τ-faithful algorithm, in a way that depends only on the description u of the bits seen so far. Note that interaction-faithfulness as defined earlier is indeed equivalent to τ-faithfulness for an appropriately-defined $\tau = \tau_{\text{int}}$.

For $k > 1$, define the *k-fold product* of restriction map τ, denoted $\tau^{\otimes k} : \{0,1\}^{kn} \to \{0,1\}^{kn}$, by $\tau^{\otimes k}(u^1, \ldots, u^k) := (\tau(u^1), \ldots, \tau(u^k))$. The map $\tau^{\otimes k}$ can be interpreted as a restriction map for algorithms making queries to a collection $x^1, \ldots, x^k$ of n-bit strings. Note that $\mathcal{R}$ is $\tau^{\otimes k}$-faithful exactly if for each $j \in [k]$, $\mathcal{R}$'s queries to the j-th input (considered alone) are always τ-faithful. Thus, the k-fold IA interaction problem defined by $(\mathcal{M}, P, \mu)$ has "difficulty" equivalent to the k-fold relation problem defined by (P_ξ, μ_ξ) for $\tau_{\text{int}}^{\otimes k}$-faithful algorithms, provided N is chosen large enough in the definition of $\xi(\cdot)$ (relative to the query bounds we are interested in).

In light of these observations, a DPT for IA interaction algorithms follows by straightforward translation from the following DPT (generalizing Theorem 2.5.2) for standard query algorithms obeying a restriction map:

Theorem 2.9.1. *Let $P \subseteq \{0,1\}^n \times B$ be a total relation such that any T-query, τ-faithful algorithm solves P with probability at most $1 - \varepsilon$ under input distribution μ.*

For any algorithm $\mathcal{R}$ making queries to inputs $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^k) \sim \mu^{\otimes k}$ and producing output in B^k, define the random set $S[\mathbf{x}]$ as in Theorem 2.5.2.

Suppose $\mathcal{R}$ is $\tau^{\otimes k}$-faithful and $\alpha\varepsilon Tk$-query-bounded for some $\alpha \in (0,1]$, and $\mathcal{A}$ is any monotone subset of $\mathcal{P}([k])$. Then conclusions 1 and 2 in Theorem 2.5.2 also hold for $\mathcal{R}$.

Proof. (Sketch) The proof follows that of Theorem 2.5.2; we only describe the differences. For $u \in \{0, 1, *\}^n$, and for a deterministic algorithm $\mathcal{D}$ on n input bits, let

$$W_P(u, \mathcal{D}) \ := \ \Pr_{\mathbf{y} \sim \mu^{(u)}}[(\mathbf{y}, \mathcal{D}(\mathbf{y})) \in P].$$

Let us say that $\mathcal{D}$ is *u-inducing* if, on any input $x \in \{0, 1\}^n$ which extends[5] u, the outcome of $\mathcal{D}$'s first $|u|$ queries to x are described by u.

If $|u| \leq T$, define $W_{P,\tau}(u) := \max_{\mathcal{D}} W_P(u, \mathcal{D})$, where the max ranges over all deterministic, u-inducing, τ-faithful algorithms $\mathcal{D}$ making at most T queries. We have:

Lemma 2.9.2. *1.* $W_{P,\tau}(*^n) \leq 1 - \varepsilon$.

2. For any $u \in \{0, 1, *\}^n$ with $|u| < T$, and any $i \in [n]$ satisfying $\tau(u)_i = 1$, we have

$$\mathbb{E}_{\mathbf{y} \sim \mu^{(u)}}[W_{P,\tau}(u[x_i \leftarrow \mathbf{y}_i])] \ \leq \ W_{P,\tau}(u).$$

The proof of Lemma 2.9.2 follows that of Lemma 2.2.2. The rest of the proof of Theorem 2.9.1 follows that of Theorem 2.5.2, with $W_{P,\tau}(u)$ taking the place of $W_P(u)$. $\square$

One can also prove a DPT for search problems for τ-faithful query algorithms, along the lines of Theorem 2.7.1. When applied to interactive automata via the translation described earlier, search problems correspond to tasks whose success conditions are defined in terms of the interaction itself (rather than the hidden seed of the IA, or any output produced by the query algorithm).

2.10 Questions for future work

1. Can the bounds in our threshold DPTs and XOR lemma be improved? For example, in Theorem 2.0.3, can one improve the success probability bound to $\frac{1}{2}\left(1 + [1 - 2\varepsilon + O(\alpha\varepsilon)]^k\right)$?

[5](as defined in Section 2.7.1)

2. It is still unknown what worst-case success probability in computing $f^{\otimes k}$ can be achieved in general, when the number of queries allowed is $\alpha R_2(f)k$ for $\alpha \gg 1$. The corresponding question in the quantum query model was settled by Buhrman *et al.* [BNRdW07]. As mentioned earlier, $O(R_2(f)k \log k)$ queries always suffice to compute $f^{\otimes k}$ with high success probability; work of Feige *et al.* [FRPU94] implies that we cannot do better than this by using a bounded-error randomized algorithm for f in a black-box fashion.

3. Can ideas from our work be helpful in obtaining new results in other computational models? For example, Lee and Roland [LR12] prove a threshold DPT for quantum query algorithms computing Boolean functions, where the query bound scales as $\Omega(Q_2(f)k)$. Can we extend this to a *generalized* threshold DPT, analogous to our Theorem 2.5.2?

2.11 Chapter acknowledgments

I thank Ronald de Wolf for numerous helpful comments, and in particular for encouraging me to look at threshold DPTs. I also thank the anonymous reviewers.

Chapter 3

Joint Complexity in the Decision Tree Model

3.0.1 Results of this chapter

In this chapter we propose and begin a systematic study of computational models from the point of view of the *diversity* of possible behaviors of their joint complexity. Formally we approach this in the following way. We fix a computational model M capable of producing output over any finite alphabet, and a notion of cost for that model (such as worst-case number of comparisons, decision tree depth, etc.). Given a collection $F = \{f_1(x), f_2(x), \ldots f_\ell(x)\}$ of total functions on a common input $x \in \{0,1\}^n$, define the *joint cost function* $C_F(X) : \{0,1\}^\ell \to \mathbb{R}$ by letting $C_F(X)$ equal the minimum cost of any algorithm in M that, on input $x \in \{0,1\}^n$, outputs in some specified order the values $f_i(x)$, for every i such that $X_i = 1$. (We use capitalized variable names for vectors that index subsets of a function family F, to distinguish them from the lower-case vectors x which will denote inputs to F.)

The question we are interested in is this: What *kinds* of functions $C_F(X)$ can arise in this way, when we range over all choices of F?

There are some obvious constraints on C_F. For many reasonable definitions of cost, C_F will be **nonnegative and integer-valued** (at least for worst-case notions of cost, which we will always be considering). As long as the functions in F are non-constant (and we will assume this throughout), $C_F(X)$ will be 0 if and only if $X = \mathbf{0}$.

We expect C_F to be **monotone** (but not necessarily strictly monotone), since any algorithm computing a subset $S \subseteq F$ of functions can be trivially modified to compute any $S' \subset S$. Finally, C_F should be **subadditive**; that is, we should always have $C_F(X \vee Y) \leq C_F(X) + C_F(Y)$. This is because an algorithm can always solve two subcollections of functions separately and then combine the results in its output.

Are there any other constraints? We now illustrate by example that, for at least some models of computation, there are functions $C(X)$ obeying the constraints above, which do not correspond to $C_F(X)$ for any choice of collection F. We consider the deterministic decision tree model, with depth as the complexity measure.

For $X \in \{0,1\}^3$, let $||X||$ be the Hamming weight of X, and define

$$C^*(X) = \begin{cases} 0 & \text{if } ||X|| = 0, \\ 1 & \text{if } ||X|| \in \{1,2\}, \\ 2 & \text{if } ||X|| = 3. \end{cases}$$

One can verify that $C^*(X)$ satisfies nonnegativity, monotonicity, and subadditivity. Now suppose for contradiction's sake that some family $F = \{f_1(x), f_2(x), f_3(x)\}$ satisfies $C_F(X) = C^*(X)$ for all X. This means that any two functions in F can be computed with one query to x, while it requires 2 queries to compute all three.

Since $C^*(1,1,0) = 1$, f_1 and f_2 must depend only on a single shared input bit x_i. Similarly $C^*(1,0,1) = 1$ implies that f_1, f_3 each depend on a single shared input bit x_j, so $i = j$. But then a single query to x_i determines all three functions, so that $C_F(1,1,1) = 1 \neq C^*(1,1,1)$. This contradicts our assumption.

The example of C^* suggests that other significant constraints might exist on joint cost functions for decision-tree complexity. However, we will show that there is a strong sense in which this is false. In Section 3.1.1 we formally define *economic cost functions* as functions obeying nonnegativity (strict except at $\mathbf{0}$), monotonicity, and subadditivity; the rest of the chapter is then devoted to proving the following result:

Theorem 3.0.1. *Given any collection*

$$F = \{f_1(x), f_2(x), \dots f_\ell(x)\}$$

of nonconstant Boolean functions, $C_F(X)$ (defined relative to the adaptive query model) is an economic cost function.

Furthermore, given any economic cost function $C(X) : \{0,1\}^\ell \to \mathbb{Z}$, and an $\varepsilon > 0$, there exist integers n, $T > 0$, and a collection $F = \{f_1(x), \dots f_\ell(x)\}$ of (total) Boolean functions on a common n-bit input x, such that, for all X,

$$(1 - \varepsilon)T \cdot C(X) \leq C_F(X) \leq (1 + \varepsilon)T \cdot C(X) .$$

That is, there exist joint cost functions $C_F(X)$ to approximate any economic cost function, if that economic cost function is allowed to be "scaled up" by a multiplicative factor and if we allow a multiplicative error of $(1 \pm \varepsilon)$. Theorem 3.0.1 would remain true if we allowed economic cost functions to take non-integral values, since (up to a scaling factor) such functions can be arbitrarily well-approximated by integral economic cost functions.

3.0.2 Comparison with Shannon entropy

It is interesting to compare our result on joint cost functions in the query model with the study of *Shannon entropy measure* $H(X)$, a key measure of the information content of a random variable X. The Shannon entropy also satisfies natural nonnegativity, monotonicity, and subadditivity properties. Here monotonicity means $H(X) \leq H(X, Y)$ for all random variables X, Y; $H(X, Y)$ is simply the entropy of the pair-variable (X, Y). Subadditivity means that $H(X, Y) \leq H(X) + H(Y)$ for all X, Y. Researchers wondered whether these basic "Shannon inequalities" imply *all* the valid inequalities that obtain universally for the joint entropies of finite collections of random variables. It turns out that this is not the case; other linear, homogeneous inequalities constrain the joint entropies of 4 or more variables. This was first discovered by Zhang and Yeung [ZY97], and the study of such inequalities has turned out to be rather complex; see, e.g., [MMRV02].[1]

By contrast, our Theorem 3.0.1 characterizes the linear, homogeneous inequalities that are universally valid for joint cost functions: they are precisely the ones that hold for all economic cost functions. By standard facts, all such inequalities can be derived as linear combinations of "basic" inequalities of the three types we have described. In this sense, the set of realizable joint cost functions is both *simpler* than the joint Shannon entropy (because it is easier to characterize in these terms), yet also *more diverse* (because it is less constrained). We find this contrast intriguing.

3.0.3 Economic cost functions, computational models, and universality

We summarize Theorem 3.0.1 by saying that the adaptive query model is *universal for economic cost functions.* For any model M of computation with an associated notion of cost, we say that M is universal for economic cost functions if the analogue of Theorem 3.0.1 is true with joint cost functions from M replacing those of the adaptive query model.

As a consequence of Theorem 3.0.1, we obtain a universality result for any deterministic, adaptive model into which we can "embed the query model." For example of what we mean, let us consider the comparison model of computation over lists of integers in which a basic step is a comparison of two list elements. Let $F = \{f_1(x), \dots f_\ell(x)\}$ be any collection of Boolean functions with domain $\{0,1\}^n$. Based on F, we define a collection $G = \{g_1(a), \dots g_\ell(a)\}$ of Boolean-valued functions $g_j(a)$ taking as common input a list of $2n$ integers $a = (a_1, \dots a_{2n})$. First, let $b_i = b_i(a)$ be an indicator variable for the event $[a_{2i-1} < a_{2i}]$. Then define

$$g_j(a) \ := \ f_j(b_1, \dots b_n) \ .$$

[1]This is not to say that there are no simple characterizations of the Shannon entropy, only that there is no known simple description of the set of universally-valid entropy inequalities.

The values b_i are each computable by a single comparison, and each pair $b_i, b_{i'}$ are functions of disjoint variable-sets, so we see that the cost of computing any subcollection of G on a common input is exactly the cost (in the Boolean adaptive-query model) of computing the corresponding subcollection of F.

Since the query model thus "embeds" into the comparison model (and since cost functions in the comparison model can be easily seen to be economic cost functions), in light of Theorem 3.0.1 we conclude:

Corollary 3.0.2. *The comparison model is universal for economic cost functions.* □

Proving such a result in the communication model seems difficult, and would require a better understanding of the "disjoint-inputs intuition" for communication (see Chapter 1, Section 1.3). We next state a "Query-Model Embedding Conjecture" that would suffice to prove that the communication model is universal for economic cost functions, along the lines of Corollary 3.0.2.

Let $n, k > 0$ be integers. Given $f(x, y) : \{0,1\}^{2n} \to \{0,1\}$, and a function $g(z) : \{0,1\}^k \to \mathbb{N}$, define a function $(g \circ f) : \{0,1\}^{2nk} \to \mathbb{N}$ by

$$(g \circ f)(x_1, y_1, x_2, y_2, \ldots x_k, y_k) \ := \ g(f(x_1, y_1,), \ldots f(x_k, y_k)) \ .$$

(This kind of composition of functions has been studied before in the communication setting, e.g., in [KRW95].) In the communication problem for $g \circ f$ we understand Alice to receive all x-inputs and Bob all y-inputs. Let $\mathrm{cc}(h)$ denote the (adaptive, deterministic) communication complexity of computing the ($\mathbb{N}$-valued) function h, by a protocol in which Alice speaks first, and both parties learn the function value. As usual let $D(g)$ denote the decision tree complexity of computing g.

Conjecture 3.0.3. *For every $k \in \mathbb{N}$ and $\delta \in (0,1)$, there exists $n > 0$ and a function $f : \{0,1\}^{2n} \to \{0,1\}$ (with $\mathrm{cc}(f) > 0$) such that for all $g : \{0,1\}^k \to \mathbb{N}$, we have*

$$\mathrm{cc}(g \circ f) \ \geq \ (1 - \delta)\, \mathrm{cc}(f) D(g) \ .$$

We can show a nearly matching upper bound

$$\mathrm{cc}(g \circ f) \;\leq\; \mathrm{cc}(f) D(g)$$

for all choices of g, f, by the following protocol idea: The players consider $\{b_i := f_i(x_i, y_i)\}_{i \leq k}$ as bits to be "queried," and simulate an optimal decision tree on these bits; whenever they want to determine some b_j, they execute the optimal communication protocol for f on (x_j, y_j). This makes them both learn $f(x_j, y_j)$, so they both know which bit b_i is to be "queried" next.

Note that the conjecture asserts a strengthened form of the disjoint-inputs intuition, for some particular family of functions f: by setting g to be a function that outputs an encoding of its input, we see that computing $f(x, y)$ on k independent input pairs requires nearly k times as much communication as for one pair.

Unable to prove the conjecture, we can at least note the following: the conjecture really is sensitive to our choice of "inner" function f. For example, let $f(x, y) = x \vee y$, and let g be the OR function on k bits. Then the communication complexity of computing $(g \circ f) = \bigvee_{i=1}^{k}(x_i \vee y_i) = (\bigvee_{i=1}^{k} x_i) \vee (\bigvee_{i=1}^{k} y_i)$ is $O(1)$, even though each $f(x_i, y_i)$ has nonzero communication complexity and the OR_k function has decision tree complexity k. We suspect, however, that a *random* function $f(x, y)$, on an input size sufficiently large compared to k and $\frac{1}{\delta}$, should be a suitable inner function for our conjecture to hold.

Our conjecture also appears somewhat related to the Enumeration and Elimination Conjectures of [ABG$^+$01] (so far unresolved; see also [BDKW10]). These are another type of variant of the disjoint-inputs intuition. We are not, however, aware of any formal implication between these conjectures and ours.

3.0.4 Outline and methods

To prove Theorem 3.0.1, a first key tool is the notion of hitting sets of weighted set systems. Given a set family $\mathbf{A} = \{A_1, \ldots A_\ell\}$ over a universe U and a weight function $w : U \to \mathbb{R}$, the weight of a subset $B \subseteq U$ is defined as the sum of B's members' weights. B is called a *hitting set* for a subfamily $S \subseteq \mathbf{A}$ if B intersects each $A_i \in S$. The *hitting-set cost function* $C_{\mathbf{A}}(X) : \{0,1\}^\ell \to \mathbb{N}$ gives the minimum weight of any B that is a hitting set for $S_X = \{A_i : X_i = 1\}$.

We use these notions to derive a useful representation lemma (Lemma 3.1.6): for any economic cost function $C(X)$ on ℓ bits, there exists a family $\mathbf{A} = \{A_1, \dots A_\ell\}$ over a weighted universe (U, w), such that $C_{\mathbf{A}}(X) = C(X)$. The simple proof of Lemma 3.1.6 is given in Section 3.1.4.

As a concrete example to illustrate the expressive power of these hitting-set cost functions, we present a simple weighted set system whose hitting-set cost function is exactly the example function $C^*(X)$ presented in Section 3.0.1. (This will not be the set system that would be produced by our general method.) Let $\mathbf{A}$ be the family of all 2-element sets over the universe $U = \{u_1, u_2, u_3\}$ (so, $|\mathbf{A}| = 3$), and let $w(u_i) = 1$, for each $u_i \in U$. Note that any one or two of the sets from $\mathbf{A}$ has a hitting set of size 1, but to hit all of $\mathbf{A}$ requires two elements. Since each element has unit weight, $C_{\mathbf{A}}(X)$ is exactly $C^*(X)$.

Returning now to the discussion of our main strategy, it will suffice to solve the following problem: given a weighted set system $\mathbf{A} = \{A_1, A_2, \dots A_\ell\}$, produce a collection $F = \{f_1, \dots f_\ell\}$ of Boolean functions over some domain $\{0,1\}^n$ such that $C_F(X)$ is approximately a multiple of $C_{\mathbf{A}}(X)$.[2]

Here is a high-level sketch of our collection F. For each $u \in U$, we create a block y_u of input variables called the "bin" for u; x is the disjoint union of these blocks. y_u represents, in a carefully defined way, the contents of a conceptual "bin" which contains at most one "key" k from a large set K called the "keyspace."

The bin representations and a value $T > 0$ are chosen in such a way that the following (informal) conditions hold:

(i) The contents of any bin y_u can be determined in at most $w(u)T$ queries;

(ii) For any fixed $k \in K$ and any bin y_u, it can be determined with "very few" queries whether k is in the bin (so that this step is "essentially free" in comparison to

[2]We remark that if F were allowed to be *partial* functions, this construction would become much easier, but would also lose most of its interest.

the queries described in (i));

(iii) If the number of queries an algorithm makes to the bin y_u is even "noticeably" less than $w(u)T$, the amount of information it gains about the bin contents is "tiny," that is, the data seen is consistent with almost any $k \in K$ occupying the bin. (At least, this outcome is unavoidable when an appropriately chosen adversary strategy determines the answers to queries as they are made.)

We will formalize bins obeying the above properties in the notion of "mystery bin functions" in Section 3.2.2.

Returning to the sketch construction of our function collection, for $i \in \{1, 2, \dots l\}$, define $f_i(x) = 1$ iff there exists some $k \in K$ that is contained in each of the "mystery bins" y_u corresponding to elements $u \in A_i$.

To informally analyze this collection, fix any nonzero $X \in \{0,1\}^\ell$, indexing a subcollection $S_X \subseteq \mathbf{A}$.

For an upper bound on $C_F(X)$, pick a minimal-weight hitting set B for S_X, so $w(B) = C_{\mathbf{A}}(X)$. In the first phase, for each $u \in B$, let our algorithm determine the bin contents of y_u. By property (i) this phase uses at most $w(B)T$ queries.

Next comes the second phase. For every $A_i \in S_X$, there's a $u \in A_i \cap B$, whose bin contents we've determined; if the bin y_u was empty we can conclude $f_i(x) = 0$. If the bin contained the element $k \in K$ (remember that at most one key lies in each bin), query the bins of all other elements $u' \in A_i$ to see if k is in all of them. If so, $f_i(x) = 1$, otherwise $f_i(x) = 0$.

Thus our algorithm succeeds in computing $\{f_i(x) : X_i = 1\}$. By property (ii) above, the query complexity of the second phase is "negligible," giving $C_F(X) \leq (1 + \varepsilon)T \cdot C_{\mathbf{A}}(X)$ as needed.

For the lower bound, we pit any algorithm using fewer than $(1-\varepsilon)T \cdot C_{\mathbf{A}}(X)$ queries against an adversary strategy that runs the adversary strategies for each mystery bin in parallel. Since $C_{\mathbf{A}}(X)$ is the minimal cost of any hitting set for S_X, at the end of this run of the algorithm there must exist some $A_i \in S_X$ such that for each $u \in A_i$, y_u receives noticeably less than $w(u)T$ queries. Using property (iii) of mystery bins, we then argue that the algorithm fails to determine the value $f_i(x)$. This will prove $C_F(X) \geq (1-\varepsilon)T \cdot C_{\mathbf{A}}(X)$.

The main technical challenge in implementing the above idea is to design the right representation of the bin contents of the blocks y_u to guarantee the "mystery bin" properties. To build mystery bin functions, we will exploit a small polynomial separation between decision tree depth and unambiguous certificate complexity, due to Savický [Sav02]. We describe his result, and reformulate it for our purposes, in Section 3.1.3.

How does Savický's result facilitate our construction of "mystery bins?" Roughly speaking, the gap between deterministic and certificate complexity in his theorem yields the query-complexity gap between properties (i) and (ii) of mystery bins, while the key contribution of unambiguity is in allowing us to construct mystery bin functions in which the bin always contains at most one key. In the algorithm described above to compute $\{f_i(x) : X_i = 1\}$, this allows the query complexity of the second phase to remain negligible, yielding the upper bound we need on $C_F(X)$.

In the course of building mystery bin functions, another useful device called a "weak exposure-resilient function" is also introduced and used. This object, an encoding method that looks uninformative when restricted to a small number of coordinates, is indeed a weak special case of the "exposure-resilient functions" studied in [CDH$^+$00]. However, the parameters we need are easily obtainable and so we provide a self-contained (probabilistic) construction and analysis.

3.1 Definitions and preliminary results

3.1.1 Vectors and economic cost functions

Given two bitvectors $X = (X_1, \ldots X_\ell), Y = (Y_1, \ldots Y_\ell)$, we write $X \leq Y$ if $X_i \leq Y_i$, for all $i = 1, 2, \ldots n$. We define the vector $Z = X \vee Y$ by the rule $Z_i = X_i \vee Y_i$.

Note that, in this chapter, we use capital-letter variable names (X, Y, Z) to refer to vectors indexing "bundles of goods," and we use lower-case variable names to refer to other vectors, such as the inputs and outputs to functions whose decision-tree complexity we will analyze.

Definition 3.1.1. *Say that a function $C(X) : \{0,1\}^\ell \to \mathbb{Z}$ is an economic cost function if it satisfies the following conditions:*

(1) $C(X) \geq 0$, and $C(X) = 0 \Leftrightarrow X = \mathbf{0}$;

(2) For all X, Y, $X \leq Y$ implies $C(X) \leq C(Y)$;

(3) For all X, Y, $C(X \vee Y) \leq C(X) + C(Y)$.

We call such functions "economic cost functions" due to the following informal interpretation: consider the input $X \in \{0,1\}^\ell$ to C represent a certain subset of ℓ distinct "goods" that a company is capable of producing. If $C(X)$ represents the cost to the company of producing one each of the goods indexed by the 1-entries of X, then intuitively, we expect C to obey condition (1) because there's "no free lunch." Condition (2) supposes that, to produce one bundle of goods, one can always produce a larger bundle of goods and "throw away" the unwanted ones (and we assume free garbage disposal). Condition (3) supposes that, to produce two (possibly overlapping) bundles X, Y of goods, we can always separately produce the two bundles. Equality may not always hold in condition (3), even for disjoint bundles of goods, due to possible "synergies" arising in production.

We note in passing that the definition of economic cost functions is a special case of the more general notion of "outer measures" on lattices; see [Bir67], Chapter 9.

3.1.2 Decision trees and joint cost functions

Recall the formal definition of decision trees from Section 2.1. By the *depth* of T, denoted $D(T)$ in this chapter, we again mean the length of the longest path from the root in T, stepping exclusively from parent to child. Given a collection of functions $S = \{f_1(x), f_2(x), \ldots f_\ell(x)\}$, we define the (deterministic, adaptive) *query complexity* of S as $D(S) = \min \{d :$ there exists a decision tree T of depth d computing the collection $S\}$. If S is a single function, $S = \{f\}$, we also write $D(f) = D(S)$.

We next define, for any finite collection F of functions, a function C_F which summarizes the joint synergies existing among the members of F (relative to the decision-tree depth model of cost). Given a collection F of functions, $F = \{f_1(x), f_2(x), \ldots f_\ell(x)\}$ on a common input, we define the *joint cost function* $C_F(X) : \{0,1\}^\ell \to \mathbb{Z}$ *associated with* F by $C_F(X) = D(S_X)$, where $f_i \in S_X \Leftrightarrow X_i = 1$. We define $C_F(\mathbf{0}) = 0$.

Thus $C_F(X)$ gives the "cost" of certain "bundles of goods," where cost is interpreted as decision tree depth, and the different "bundles of goods" in question are the various subcollections of functions from F. As promised by part of Theorem 3.0.1, we will show (Lemma 3.2.1) that for any F, $C_F(X)$ is always an economic cost function as defined in Section 3.1.1.

3.1.3 Search problems and TUSPs

Although in this chapter we are primarily interested in the query complexity of (collections of) decision problems, our proof techniques also involve *search problems* (in the query model), defined next.

As in Section 2.7.1, say that a string $w \in \{0,1,*\}^n$ *agrees with* $x \in \{0,1\}^n$ if for all $i \in [n]$, $w_i \in \{0,1\}$ implies $w_i = x_i$. Also as before, a search problem on domain $\{0,1\}^n$ is specified by a subset $W \subseteq \{0,1,*\}^n$ called the "witnesses." We say that a decision tree T solves the search problem W if (i) for every input x that agrees with at least one $w \in W$, $T(x)$ outputs some $w' \in W$ agreeing with x (if there are more than one such w', we don't care which one), and (ii) if x agrees with no $w \in W$, $T(x)$ outputs "no match." Note here the slight difference in our definition of *solving* a search problem, compared with Chapter 2.

Given a search problem W, let $s(W)$ denote the maximum number of 0/1 entries in any $w \in W$. Write $D(W)$ to denote the minimum depth of any decision tree solving W.

W is called a *total* search problem if all $x \in \{0,1\}^n$ agree with at least one $w \in W$. W is called a *unique* search problem if all x agree with at most one $w \in W$. In this chapter we will deal with search problems W that are both total and unique; we call such a W a *TUSP* for brevity. A TUSP W defines a (total, single-valued) function from $\{0,1\}^n \to W$ mapping x to the unique witness w agreeing with x; we denote this function by $W(x)$.

For TUSPs W, as for other search problems, it is easy to see that $s(W) \leq D(W)$: in any decision tree T solving W, the variables read by T on an input x must always include all the 0/1 entries in $w = W(x)$. In fact, up to an at-most quadratic factor, this inequality is tight:

Theorem 3.1.2. *[BI87], [HH91], [Tar89] For all unique search problems, $D(W) \leq s(W)^2$.*

Proof. The proof is essentially identical to that of a related result, which states that decision-tree depth complexity is most the square of the "certificate complexity" for Boolean functions [BI87], [HH91], [Tar89].

Let $s = s(W)$. We define a query algorithm as follows: on input x, proceed in phases. At the beginning of phase t, let $W_t \subseteq W$ be the set of "live" witnesses, i.e. those that agree with the bits of x seen so far. If $W_t = \{w\}$, then the algorithm outputs w. Otherwise, say that $i \in [n]$ is an "active" coordinate for $w \in W_t$ if $w_i \in \{0,1\}$ and x_i has not been queried. In each phase t, the algorithm picks an arbitrary $w \in W_t$ and queries x on each of the active coordinates i for w.

Since W is a unique search problem, every distinct $w, w' \in W_t$ disagree on at least one coordinate i active for both w and w'. Thus, in each phase t and for every $w \in W_t$, the number of active coordinates for w decreases by at least one. After at most s phases, then, no live w has any active coordinates; hence such a w either disagrees with x on one of the bits already seen, or agrees with x on each i with $w_i \in \{0,1\}$. As W is total, it follows that the decision tree for our algorithm solves W, while making at most $s + (s-1) + \ldots + 1 \leq s^2$ queries. $\square$

Savický [Sav02] proved a theorem implying that, in general, $D(W)$ can be polynomially larger than $s(W)$ for TUSPs. He uses different terminology and states a slightly different result than we need, so we will have to "unpack" his result a little.

A *DNF formula* ψ is an OR of clauses, each of which consists of the AND of one or more literals or negated literals. Say that ψ is an *unambiguous DNF (uDNF)* if any input x satisfies at most one of its clauses. Savický showed

Theorem 3.1.3. *[Sav02] There exists a family of functions*

$$\{G_i : \{0,1\}^{4^i} \to \{0,1\}\}_{i\in\mathbb{N}} \ , \qquad \textit{such that}$$

(i) G_i and $\overline{G_i}$ each have uDNF representations in which each clause has size at most $s_i = 3^i$;

(ii) $D(G_i) \geq \frac{4^i+2}{3} = \Omega(s_i^{\gamma})$, where $\gamma = \log_3(4) > 1$.

Theorem 3.1.3 is very close, but not identical, to the combination of Theorems 3.1 and 3.6 from [Sav02]. That paper was concerned with the complexity measure $p(f)$ defined as the minimal number of clauses in any uDNF representation of f, whereas we are concerned with minimizing the maximum size of any clause as in Theorem 3.1.3; also, Savický lower-bounds the number of leaves of any decision tree for f rather than its depth. However, the particular function family [Sav02] gives is seen by inspection (and noted in [Sav02]) to satisfy condition (i), while condition (ii) follows from Savický's lower bound on number of leaves in any decision tree computing G_i, after noting that a decision tree with k leaves has depth at least $\lceil \log_2(k) \rceil$. this yields Theorem 3.1.3.

We remark that it to prove our main theorem, we don't really need the full strength of Theorem 3.1.3. Specifically, it would be enough that just *one* of G_i or $\overline{G_i}$ had uDNF representations with short clauses relative to the query complexity (or even short-clause DNF representations with a bounded number of satisfied clauses per input). However, using the full statement of Theorem 3.1.3 makes our proof slightly simpler.

We can derive from Theorem 3.1.3 the following form of Savický's result, which will be more convenient for us:

Theorem 3.1.4. *There exists a family of TUSPs*

$$\{W_N \ \subset \ \{0,1,*\}^{m(N)}\}_{N=1}^{\infty}$$

on $m(N) \leq \mathrm{poly}(N)$ *input bits, and a constant* $\alpha > 0$*, such that* $D(W_N) \geq s(W_N)^{1+\alpha}$*, while* $s(W_N) \geq N$*.*

Proof. For any $i > 0$, given uDNF representations F_1, F_2 of G_i and $\overline{G_i}$ respectively satisfying condition (i) of Theorem 3.1.3, we define a search problem V_i: For every clause c in one of the F_i's, define a witness $w_c \in V_i$ that has 0/1 entries exactly on the variables contained in c, with these variables set in the unique way satisfying c (remember c is a conjunction). From the facts that F_1, F_2 are each uDNFs and that every input x satisfies exactly one of them, we conclude that V_i is a TUSP.

By condition (i) of Theorem 3.1.3, $s(V_i) \leq 3^i$. On the other hand, since any decision tree for V_i immediately yields a decision tree of the same depth for G_i, we have

$$D(V_i) \ \geq \ \frac{4^i + 2}{3} > \frac{(3^i)^{\log_3(4)}}{3} \ ,$$

which for large enough i is greater than $s(V_i)^{1+\alpha}$ for an appropriate constant $\alpha > 0$. Also, by Theorems 3.1.2 and 3.1.3,

$$s(V_i) \ \geq \ \sqrt{D(V_i)} > \frac{2^i}{\sqrt{3}} \ .$$

Now we simply set $W_N := V_{\lceil \log(N) \rceil + 1}$. We verify that $m(N) = 4^{\lceil \log(N) \rceil + 1} \leq \mathrm{poly}(N)$. □

In order to make effective use of the decision-tree depth lower bound contained in Theorem 3.1.4, we will need the following folklore result, showing the optimality of the "adversary method" in decision tree complexity:

Claim 3.1.5. *Let* B *be a finite set. Suppose* $f(x) : \{0,1\}^n \to B$ *satisfies* $D(f) \geq t > 0$*; then there exists an adversary strategy for determining the bits of* x *as they are queried (depending only on the sequence of queries made so far), such that for any query strategy making* $(t-1)$ *queries to* x*, the bits of* x *fixed in the process do not uniquely determine the value of* $f(x)$*.*

The proof of Claim 3.1.5 is a simple proof by induction, and is omitted. Note that Claim 3.1.5 applies in particular when $f(x) = W(x)$ is the (total, single-valued) function associated with a TUSP W.

3.1.4 Set systems and hitting sets

As a final preliminary definition, we introduce hitting sets of set systems. Given a finite universe U, and a collection $\mathbf{A} = \{A_1, A_2, \dots A_\ell\}$ of subsets of U, we say a set $B \subseteq U$ *hits* $\mathbf{A}$, or is a *hitting set for* $\mathbf{A}$, if $B \cap A_i \neq \emptyset$ for all $i \leq \ell$.

Given a positive function $w : U \to \mathbb{N}$ called a "weight function," define the weight of a set $A \subseteq U$ as $w(A) = \Sigma_{u \in A} w(u)$. Define the *weighted hitting set cost* of the collection $\mathbf{A}$ (relative to w) as $\rho(\mathbf{A}) = \min \{c :$ there exists a hitting set $B \subseteq U$ for $\mathbf{A}$ with $w(B) \leq c\}$.

For a collection $\mathbf{A} = \{A_1, \dots A_\ell\}$, and given $X \in \{0,1\}^\ell$, define $S_X = \{A_i : X_i = 1\}$. Define the *weighted hitting set cost function* $C_{\mathbf{A}}(X) : \{0,1\}^\ell \to \mathbb{N}$ by $C_{\mathbf{A}}(X) = \rho(S_X)$.

We now prove that the class of weighted hitting set cost functions is *exactly* the class of economic cost functions.

Lemma 3.1.6. *For any set system* $\mathbf{A}$ *and weight function* w*,* $C_{\mathbf{A}}$ *is an economic cost function. Moreover, given any economic cost function* $C(X) : \{0,1\}^\ell \to \mathbb{N}$*, there exists a finite set* U *and a collection* $\mathbf{A} = \{A_1, \dots A_\ell\}$ *of subsets of* U*, such that for all* $X \in \{0,1\}^\ell$*,* $C_{\mathbf{A}}(X) = C(X)$*.*

Proof. First we show that $C_{\mathbf{A}}$ is always an economic cost function. That condition (1) of the definition of economic cost functions is satisfied is immediate. For condition (2), note that if $X \leq Y$, $S_X \subseteq S_Y$, so any hitting set for S_Y is also one for S_X. Thus $C_{\mathbf{A}}(X) = \rho(S_X) \leq \rho(S_Y) = C_{\mathbf{A}}(Y)$, as needed.

To see that condition (3) is satisfied, note that if B_X, B_Y are hitting sets for S_X, S_Y, then $B_X \cup B_Y$ is a hitting set for $S_X \cup S_Y = S_{X \vee Y}$, and $w(B_X \cup B_Y) \leq w(B_X) + w(B_Y)$.

For the second part, let $C(X) : \{0,1\}^\ell \to \mathbb{N}$ be an economic cost function. We define a set system and weight function as follows. Let U be a set of size 2^ℓ, indexed by ℓ-bit vectors as $U := \{b_X : X \in \{0,1\}^\ell\}$.

Let $\mathbf{A} = \{A_1, \dots A_\ell\}$, where $A_i := \{b_X : X_i = 1\}$. Finally, define $w(b_X) := C(X)$.

We claim that, for all $X = (X_1, \dots X_\ell)$, $C_{\mathbf{A}}(X) = C(X)$. First we argue that $C_{\mathbf{A}}(X) \leq C(X)$. Consider the singleton set $B = \{b_X\}$. For every i such that $X_i = 1$, $b_X \in A_i$. Thus, B is a hitting set for $S_X = \{A_i : X_i = 1\}$. By definition, then, $C_{\mathbf{A}}(X) \leq w(B) = w(b_X) = C(X)$.

Now examine any hitting set B' for $\{A_i : X_i = 1\}$, say $B' = \{b_{Z_{[j]}} : Z_{[j]} \in I \subseteq \{0,1\}^\ell\}$. For each i such that $X_i = 1$, A_i is hit by B', so there exists some $Z_{[j]} \in B'$ such that $b_{Z_{[j]}} \in A_i$. Then by definition of A_i, $Z_{[j](i)} = 1$. Thus $X \leq \bigvee_{Z_{[j]} \in I} Z_{[j]}$, and

$$w(B') \;=\; \sum_{Z_{[j]} \in I} w(b_{Z_{[j]}}) \;=\; \sum_{Z_{[j]} \in I} C(Z_{[j]}) \;\geq\; C(\bigvee_{Z_{[j]} \in I} Z_{[j]}) \;.$$

(The last inequality above holds by iterated application of property (3) of economic cost functions.) This is $\geq C(X)$, since $X \leq \bigvee_{Z_{[j]} \in I} Z_{[j]}$, and using property (2) of economic cost functions. Thus $C_{\mathbf{A}}(X) = C(X)$, as claimed. □

3.2 Proof of Theorem 3.0.1

3.2.1 First steps

The first half of Theorem 3.0.1 is easy, and recorded in Lemma 3.2.1:

Lemma 3.2.1. *If $F = \{f_1(x), f_2(x), \dots f_\ell(x)\}$ is a collection of nonconstant functions, $C_F(X)$ is an economic cost function.*

Proof. Clearly F satisfies condition (1) in the definition of economic cost functions, since $C_F(\mathbf{0}) = 0$ and all decision trees computing a nonconstant function or functions has depth at least 1.

$C_F(X)$ satisfies condition (2) since, given an optimal decision tree T for computing a collection $S = S_X$ of functions from S, and given a subset $S' = S_{X'} \subseteq S$, we can modify T by removing the coordinates of its output vectors corresponding to the functions in $S \setminus S'$, yielding a decision tree T' of the same depth computing the collection S'. So $D(S_{X'}) \leq D(S_X)$ and $C_F(X') \leq C_F(X)$.

To show that $C_F(X)$ satisfies condition (3), let T_X, T_Y be optimal decision trees of depths d_1, d_2 respectively, for computing the collections X, Y respectively. We define a decision tree T' as follows: we replace each output node u of T_X with a copy $T_{Y,u}$ of T_Y, and on an output node v of the copy $T_{Y,u}$ we place the label $(z(u), z(v))$, where $z(u)$ is the label of u in T_X and $z(v)$ is the label of v in T_Y. Then T' computes the collection $S_X \cup S_Y$ (possibly with redundant coordinates that we can remove, and up to a reordering of the outputs). The depth of the new tree is $d_1 + d_2$. This yields condition (3). $\square$

Now we turn to the second, harder half of Theorem 3.0.1. Following Lemma 3.1.6 showing the "universality" of hitting set cost functions, our approach to proving Theorem 3.0.1 is to build a collection of functions mimicking the structure of a given set system $\mathbf{A}$, where each f_i we create will correspond to some $A_i \in \mathbf{A}$. We will prove:

Lemma 3.2.2. *Given any hitting set cost function $C_{\mathbf{A}}(X) : \{0,1\}^\ell \to \mathbb{N}$ and $\varepsilon > 0$, there exist integers n, T, and a collection $F = \{f_1(x), \ldots f_\ell(x)\}$ of functions on n bits, such that, for all $X \in \{0,1\}^\ell$,*

$$C_{\mathbf{A}}(X) \cdot T(1-\varepsilon) \;\leq\; C_F(X) \;\leq\; C_{\mathbf{A}}(X) \cdot T(1+\varepsilon) \;.$$

In light of Lemmas 3.1.6 and 3.2.1, this will prove Theorem 3.0.1.

3.2.2 Bins and mystery bins

Central to our construction of the function family of Lemma 3.2.2 is a technical device called a "bin."

Definition 3.2.3. *A bin function is a function $\mathbf{B}(y)$ mapping a Boolean input y (of some fixed length) to subsets of size 0 or 1 of a set $K = \{k_1, \ldots k_M\}$ called the "keyspace." We call the input y a "bin," and say that k is "in the bin y" if $\mathbf{B}(y) = \{k\}$.*

Our input x to the function collection of Lemma 3.2.2 is going consist of disjoint bins, one bin corresponding to each $u \in U$ from our set system $\mathbf{A}$. The bins will have different parameters; loosely speaking, we want the difficulty of determining the bin contents $\mathbf{B}_u(y_u)$ of the bin y_u corresponding to $u \in U$ to be proportional to $w(u)$. This property by itself would be relatively easy to guarantee, but we need our bins to have some other special properties as well, formalized next in the definition of "mystery bins."

Definition 3.2.4. *Given $\beta \in [0, 1]$ and an integer $q \geq 1$, say that $\mathbf{B}$ has security β for q queries, and write* $\sec(\mathbf{B}, q) \geq \beta$*, if there exists an adversary strategy for answering queries to the vector y such that, for any query strategy making q queries to y, there exists a set $H \subset K$ of size $\beta|K|$, such that for any key $k \in H$, the bits of y fixed in the process are consistent with the condition $\mathbf{B}(y) = \{k\}$. (We do not require that the bits seen be consistent with the condition $\mathbf{B}(y) = \emptyset$, although the adversaries we will define in our construction do achieve this.)*

Note that in this definition, we require an adversary strategy for deciding the input bits as they are queried, with answers depending only on the questions and answers so far, *not* on the strategy/program making the queries.

Definition 3.2.5. *Fix $T > 0$, $\delta \in (0, 1)$. A bin function $\mathbf{B}(y)$ is called a (T, δ)-mystery bin function (MBF) (with keyspace K), if*

(i) There is a T-query algorithm to compute $\mathbf{B}(y)$;

(ii) For any $k \in K$, it can be decided in δT queries whether $k \in \mathbf{B}(y)$;

(iii) $\sec(\mathbf{B}, (1 - \delta)T) \geq (1 - \delta)$.

(Note the correspondence between the conditions in the definition above and their informal versions in the proof sketch from Section 3.0.1, when δ is close to 0.)

Constructing mystery bin functions seems to crucially rely on a result like Theorem 3.1.4 and its associated TUSP. Note that mystery bin functions behave quite similarly to the TUSPs from Theorem 3.1.4: given a particular potential witness $w \in W$, it is easily checked if the input x agrees with w; but computing $W(x)$ may be much harder. The main additional ingredient in mystery bin functions is the property (iii) above, which imposes on algorithms a "sharp transition" between near-total ignorance and certainty as they attempt to determine a bin's contents. This sharp transition is what will allow us to tightly analyze the function collections we will build to prove Lemma 3.2.2.

Our construction of mystery bin functions is given by the following Lemma:

Lemma 3.2.6. *For all $\delta > 0$, we can find $T, M > 0$ such that, for every integer $c \geq 1$, there exists a (cT, δ)-mystery bin function with keyspace $K = [M]$.*

3.2.3 Application of mystery bins

Before proving Lemma 3.2.6, we show how it is used to prove Lemma 3.2.2 and, hence, Theorem 3.0.1.

Say we are given a collection $\mathbf{A} = \{A_1, A_2, \dots A_\ell\}$ of subsets of a universe U, and a weight function $w : U \to \mathbb{N}$. We wish to produce a collection $F = (f_1, \dots f_\ell)$ of functions such that the cost of computing a subset of the functions of F is approximately a fixed scalar multiple of the minimum cost under w of a hitting set for the corresponding sets in $\mathbf{A}$.

Let w_{max} be the largest value of $w(u)$ over U. For each $u \in U$, we define a block of input y_u corresponding to u and a bin function $\mathbf{B}_u$ taking y_u as input. $\mathbf{B}_u$ is chosen as a $\left(w(u)T, \frac{\varepsilon}{w_{max} l |U|}\right)$-MBF with keyspace $K = [M]$, for some $T, M > 0$ independent of u, as guaranteed by Lemma 3.2.6. Let the input x to F be defined as the disjoint union of all the y_u.

For $i \leq \ell$, define $f_i(x)$ by

$$f_i(x) \;:=\; 1 \iff \exists k \in [M] \text{ such that } \mathbf{B}_u(y_u) = \{k\},\ \forall u \in A_i \;.$$

We claim that F satisfies the conclusions of Lemma 3.2.2. If $X = \mathbf{0}$ the statement is trivial, so assume $X \neq \mathbf{0}$. First we show the upper bound on $C_F(X)$. Given the corresponding nonempty subset $S_X \subseteq \mathbf{A}$, let $B \subseteq U$ be a hitting set for S_X of minimal cost:

$$w(B) \;=\; \rho(S_X) \;=\; C_{\mathbf{A}}(X) \;.$$

Define an algorithm P_X to compute $\{f_i(x) : X_i = 1\}$ as follows:

Phase 1: For each $u \in B$, compute the bin contents $\mathbf{B}_u(y_u)$.

Phase 2: For every i such that $X_i = 1$, pick some $u \in B \cap A_i$ (such a u must exist, since B is a hitting set for S_X). If in Phase 1 it was found that $\mathbf{B}_u(y_u) = \emptyset$, clearly $f_i(x) = 0$, so output 0. Otherwise, suppose $\mathbf{B}_u(y_u) = \{k\}$ for some $k \in M$; in this case, query each mystery bin $\mathbf{B}_{u'}(y_{u'})$ such that $u' \in A_i$, to ask whether $k \in \mathbf{B}_{u'}(y_{u'})$. By the definitions, $f_i(x) = 1$ iff k is indeed the contents of all such bins, so the queries of P_X determine $f_i(x)$ and the output nodes of P_X can be labeled to compute $f_i(x)$, for every i with $X_i = 1$.

It is clear that P_X computes the desired function collection. Now we bound the number of queries made by P_X. In Phase 1, each individual bin contents $\mathbf{B}_u(y_u)$ can be computed in $w(u)T$ queries, by property (i) of MBFs and the definition of $\mathbf{B}_u(y_u)$. Then altogether, at most $w(B)T = C_{\mathbf{A}}(X)T$ queries are made in this Phase.

In Phase 2, each question to a bin $y_{u'}$ asking if some k is in $\mathbf{B}_{u'}(y_{u'})$ can be answered in at most

$$\frac{\varepsilon}{w_{max} l |U|} \cdot (w(u')T) \;\leq\; \frac{\varepsilon T}{l|U|}$$

queries, using property (ii) of MBFs. Since at most $l|U|$ such questions are asked (ranging over (i, u')), in total at most εT such queries are made during Phase 2. Summing over the two Phases shows that $C_F(X) \leq D(P_X) \leq (1+\varepsilon)T \cdot C_{\mathbf{A}}(X)$, as needed.

Now we show that $C_F(X) \geq (1-\varepsilon)T \cdot C_{\mathbf{A}}(X)$, again assuming $X \neq \mathbf{0}$. We give an adversary strategy to determine the bits of x as they are queried, namely: For each $u \in U$, fix bits of y_u as they're queried, by following the adversary strategy for $\mathbf{B}_u(y_u)$ given by property (iii) in the definition of MBFs (using that $\mathbf{B}_u$ is a $\left(w(u)T, \frac{\varepsilon}{w_{max} l |U|}\right)$-MBF), and answer queries to y_u arbitrarily if this bin receives more queries than the adversary strategy for $\mathbf{B}_u(y_u)$ is guaranteed to handle.

Let P be any algorithm making fewer than $(1-\varepsilon)T \cdot C_{\mathbf{A}}(X)$ queries to the input x; we will show that the queries made by P against the adversary just defined fail to determine some value $f_i(x)$, for some i such that $X_i = 1$.

For $u \in U$, let q_u be the number of queries made by P to y_u against this adversary strategy. Let $B_P = \{u : q_u \geq (1 - \frac{\varepsilon}{2})w(u)T\}$. We claim B_P is not a hitting set for S_X. To see this, note that

$$\begin{aligned} w(B_P) &= \sum_{u \in B_P} w(u) \\ &\leq \sum_{u \in B_P} \frac{q_u}{(1 - \frac{\varepsilon}{2})T} \\ &\leq \frac{1}{(1 - \frac{\varepsilon}{2})T} \cdot \sum_{u \in U} q_u \\ &< \frac{1}{(1 - \frac{\varepsilon}{2})T} \left((1 - \varepsilon)T \cdot C_{\mathbf{A}}(X)\right) \\ &< C_{\mathbf{A}}(X) \ , \end{aligned}$$

so, by definition of $C_{\mathbf{A}}(X) = \rho(S_X)$, B_P is not a hitting set for S_X.

Thus there exists an i such that $X_i = 1$ and such that for every $u \in A_i$, $q_u < (1 - \frac{\varepsilon}{2})w(u)T$. For each such u, by the guarantee of the adversary strategy used for bin y_u, there exist at least

$$(1 - \frac{\varepsilon}{w_{max} l |U|})M \ > \ (1 - \frac{1}{|A_i|})M$$

distinct keys $k \in [M]$, such that it is consistent with the bits of y_u seen by P that $\mathbf{B}_u(y_u) = \{k\}$.

By a union bound, there exists some fixed $k \in K$ such that it is consistent with the bits seen that $\mathbf{B}_u(y_u) = \{k\}$ for *all* $u \in A_i$, which would cause $f_i(x) = 1$. On the other hand, it is also clearly consistent that *not* all such bin contents $\mathbf{B}_u(y_u)$ are equal, and hence that $f_i(x) = 0$. Thus P fails to correctly compute $f_i(x)$, for at least one input x. Since $X_i = 1$, we have shown that $C_F(X) \geq (1 - \varepsilon)T \cdot C_{\mathbf{A}}(X)$. This finishes the proof of Lemma 3.2.2, assuming Lemma 3.2.6. $\square$

3.2.4 Construction of mystery bins

Our goal in this section is to prove Lemma 3.2.6. First, suppose we can prove Lemma 3.2.6 for $c = 1$; we'll show the conclusion then follows for every $c \in \mathbb{N}$, with the same values of T and $M = |K|$.

Let $\mathbf{B}(y)$ be a (T, δ)-MBF. Say the input y has length m; define a new bin function $\mathbf{B}_c(y')$ on input $\{0, 1\}^{cm}$ with the same keyspace K by breaking the input y' into m blocks of size c, defining z_i to be the sum mod 2 of the ith block ($i \leq m$), and setting $\mathbf{B}_c(y') := \mathbf{B}(z_1, \ldots z_m)$.

The adversary strategy S' for $\mathbf{B}_c(y')$ is simply lifted from the strategy S for $\mathbf{B}(y)$, by answering queries in any given block i of y' as 0s until the last, "critical" query to that ith block is made, then answering this query as the strategy S would fix y_i conditioned on the "critical" responses made so far. Clearly any algorithm making q queries can induce at most $\lfloor \frac{q}{c} \rfloor$ critical responses from the adversary, and so property (iii) in the definition of MBFs is easily seen to be inherited by $\mathbf{B}_c$.

Similarly, any algorithm for determining the bin contents $\mathbf{B}(y)$, or for querying whether $k \in \mathbf{B}(y)$ for some $k \in K$, can be adapted to $\mathbf{B}_c$ by simply querying entire blocks at a time. This increases the number of queries by a factor c, giving properties (i) and (ii). Thus $\mathbf{B}_c(y')$ is a (cT, δ)-MBF with keyspace $[M]$, as needed.

Now we prove Lemma 3.2.6 for the case $c = 1$.

Let $N > 0$ be a (large) integer to be determined later, and let $W = W_N$ be the TUSP guaranteed by Theorem 3.1.4 for parameter N, with input size $m(N) \leq \text{poly}(N)$. For brevity write $D_N = D(W), s_N = s(W)$, and recall $D_N \geq s_N^{1+\alpha}$, $s_N \geq N$. We let $K := [D_N^2]$ be the keyspace.

We next describe the structure of the "bin" input y. y is broken into three disjoint parts, written as

$$y \ = \ (x, \mathsf{WtK}, \mathsf{KtW}) \ , \text{ where:}$$

- x will be an input to the TUSP $W = W_N$ (so $|x| = m(N)$);
- WtK, called the "witness-to-key table," will be an encoding of a function $G_{\mathsf{WtK}} : W \to K$ (with a specific encoding method to be described shortly);

- KtW, called the "key-to-witness table," will be an encoding of a function $G_{\mathsf{KtW}} : K \to W$ (with a different encoding method, also described shortly).

In our definitions, every setting to the input tables $\mathsf{WtK}, \mathsf{KtW}$ will define functions $G_{\mathsf{WtK}}, G_{\mathsf{KtW}}$ as above, and such functions will generally not have unique encodings.

Assuming for now that the two encoding schemes have been fixed, we define the bin function $\mathbf{B}(y)$ as follows: $k \in \mathbf{B}(y)$ if the witness $w = W(x)$ satisfies

$$G_{\mathsf{WtK}}(w) \ = \ k \ , \quad G_{\mathsf{KtW}}(k) \ = \ w \ .$$

Note that at most one key can be in the bin by this definition (or the bin may be empty).

Now we describe the encodings. KtW simply uses any efficient encoding with a table entry $\mathsf{KtW}|_k$ corresponding to each element k of the domain K. Since each $w \in W \subset \{0, 1, *\}^{m(N)}$ has at most s_N 0/1 entries, $|W|$ cannot be too large, namely

$$|W| \ \leq \ \sum_{i \leq s_N} \binom{m(N)}{i} \ = \ N^{O(s_N)} \ ,$$

since $m(N) \leq \mathrm{poly}(N)$. Thus each table entry $\mathsf{KtW}|_k$ in KtW can be represented using $O(s_N \log(N))$ bits. We do so, assigning "leftover" codewords arbitrarily to elements of W, so that every table defines a function (and also every function is representable).

For the encoding WtK, we want table entries to be "obfuscated," so that it takes many queries to learn anything about an individual value of G_{WtK}. We make the following definition, which resembles more-demanding definitions in [CDH+00]:

Definition 3.2.7. *Fix integers $m, d, t > 0$. Say that a mapping $J : \{0, 1\}^m \to [d]$ is an (m, d, t)-weak Exposure-Resilient Function (wERF) if for every $c \in [d]$ and every subset $S \subset [m]$ of size at most t, there is a $b \in \{0, 1\}^m$ with $J(b) = c$, and such that the entries of b indexed by S are all-zero.*

Claim 3.2.8. *For sufficiently large $N > 0$, there exists a $(\lfloor s_N^{1+\alpha/2} \rfloor, D_N^2, \lfloor \frac{1}{2} s_N^{1+\alpha/2} \rfloor)$-wERF J.*

Proof. Let J be a uniformly chosen random function from the domain $\{0,1\}^m$ (with $m = \lfloor s_N^{1+\alpha/2} \rfloor$) to the range $[d] = [D_N^2]$. We show that with nonzero probability J satisfies the definition of an (m,d,t)-wERF with $t := \lfloor \frac{1}{2} s_N^{1+\alpha/2} \rfloor$.

Fix any subset $S \subset [m]$ of size t, and a $c \in [d]$. We analyze the probability $p_{S,c}$ that there is no $b \in J^{-1}(c)$ such that b is all-zero when restricted to the coordinates in S. This is simply $(1 - \frac{1}{d})^{2^{m-t}}$. Now by our settings, for sufficiently large N we have $2^{m-t} \geq d2^{m/3}$. Thus for such N,

$$p_{S,c} \ \leq \ \left(1 - \frac{1}{d}\right)^{d2^{m/3}} \ \leq \ e^{-2^{m/3}} \ .$$

Taking a union bound over all choices of S, c, the probability that J fails to be an (m,d,t)-wERF is, for large enough N, less than $2^m d e^{-2^{m/3}} = o(1)$. So, with nonzero probability we succeed. □

Recall that in our setting $K = [D_N^2]$. We let each table entry $\mathsf{WtK}|_w$ of WtK (with position indexed by a witness $w \in W$) contain $\lfloor s_N^{1+\alpha/2} \rfloor$ bits, and define

$$G_{\mathsf{WtK}}(w) \ := \ J(\mathsf{WtK}|_w) \ ,$$

where J is as given by Claim 3.2.8.

This completes our description of the bin function $\mathbf{B}(y)$. We now show that for a large enough choice of N it is a $((1 + \frac{\delta}{2})D_N, \delta)$ mystery bin function.

First we verify property (i) in the definition of MBFs. In order for a query algorithm to determine the bin contents $\mathbf{B}(y)$, it suffices to do the following: Inspect x to determine $w = W(x)$; look up $G_{\mathsf{WtK}}(w)$, finding some key k; finally, check to see if $G_{\mathsf{KtW}}(k) = w$. If so, $\mathbf{B}(y) = \{k\}$, otherwise the bin is empty.

The first step can be implemented in D_N queries to x. For the second step, table entries of WtK are of size $\lfloor s_N^{1+\alpha/2} \rfloor$, which is $o(D_N)$ since $D_N \geq s_N^{1+\alpha}$. The third step, querying a table entry of KtW, takes $O(s_N \log(N))$ queries, which is also $o(D_N)$. Thus the total number of queries is $D_N(1 + o(1))$, less than $(1 + \frac{\delta}{2})D_N$ for large enough N. This shows property (i).

For property (ii) of MBFs, let $k \in K$ be any key; to determine if $\{k\} = \mathbf{B}(y)$, our algorithm queries $\mathsf{KtW}|_k$ to find $w = G_{\mathsf{KtW}}(k)$ and, subsequently, queries $\mathsf{WtK}|_w$ to determine if $k = G_{\mathsf{WtK}}(w)$. If not, then $\{k\} \neq \mathbf{B}(y)$, and the algorithm reports this. If $k = G_{\mathsf{WtK}}(w)$, then the algorithm makes at most $s_N = o(D_N)$ queries to x to see if x agrees with w. Note that each step takes $o(D_N)$ queries, smaller than $\delta(1 + \frac{\delta}{2})D_N$ for large N. This gives property (ii).

Finally, we show property (iii). This is the property for which we will use the fact that $|K|$ is large and entries of WtK are"exposure-resilient." Our adversary strategy against algorithms making at most $(1 - \delta)(1 + \frac{\delta}{2})D_N < (1 - \frac{\delta}{2})D_N$ queries to $y = (x, \mathsf{WtK}, \mathsf{KtW})$ is as follows:

- Answer queries to x according to a strategy, guaranteed to exist by Claim 3.1.5, that prevents any query strategy making fewer than D_N queries to x from uniquely determining the value $W(x)$. Answer all queries to $\mathsf{WtK}, \mathsf{KtW}$ with zeros.

Our proof of correctness is by contradiction. Suppose some deterministic algorithm P makes at most $(1 - \frac{\delta}{2})D_N$ queries to y against this adversary, and afterwards outputs a list L of fewer than $(1 - \delta)|K|$ keys, such that the bin contents $\mathbf{B}(y)$ is forced by the bits seen to either be empty or contain a key from L.

Define a new algorithm P' as follows: in Phase 1 P' first simulates P on y, making all the queries P does. After P terminates, define $V \subseteq W$ as the set of all witnesses w for which P has made more than $\lfloor \frac{1}{2} s_N^{1+\alpha/2} \rfloor$ queries to the table entry $\mathsf{WtK}|_w$ in WtK. In Phase 2, for each $w \in V$ in turn, P' makes any additional queries to x necessary to determine whether x agrees with w.

Say this latter set of queries in Phase 2 are "on behalf of w." Note that for every $w \in V$, at most s_N queries are made on behalf of w in Phase 2, while more than $\lfloor \frac{1}{2} s_N^{1+\alpha/2} \rfloor$ queries are made to the table entry $\mathsf{WtK}|_w$ in Phase 1. It follows that only an $o(1)$ fraction of the queries of P' are made in Phase 2, so for large enough N, P' makes fewer than D_N queries to y.

But we claim that P' succeeds in determining $W(x)$, contrary to the guarantee of our adversary strategy from Claim 3.1.5. First, after the simulated operation of P by P', say that a witness w is "live" if the bits of x seen are consistent with the possibility $W(x) = w$. Note that if there is a live witness w whose table entry in WtK has been queried at most $\lfloor \frac{1}{2} s_N^{1+\alpha/2} \rfloor$ times, then the value $G_{\mathsf{WtK}}(w)$ is completely undetermined (any value is consistent with the bits seen), since the adversary answered those queries with zeros and the function J used in defining $G_{\mathsf{WtK}}(w)$ is a $(\lfloor s_N^{1+\alpha/2} \rfloor, D_N^2, \lfloor \frac{1}{2} s_N^{1+\alpha/2} \rfloor)$-wERF.

Thus, for any key k whose table entry in KtW was not queried by P, it is consistent with the bits of y seen that $\mathbf{B}(y) = \{k\}$. Since $|K| = D_N^2 = \omega(D_N)$, if N is sufficiently large then P cannot query a bit from even a δ fraction of KtW's table entries. Hence, for P to output the list of candidates $L \subset K$ with $|L| < (1 - \delta)|K|$, it must be that for every $w \in W$ still live after the operation of P, the WtK entry for w must have been queried more than $\lfloor \frac{1}{2} s_N^{1+\alpha/2} \rfloor$ times, and thus $w \in V$.

No two distinct $w \in W$ are compatible, so it follows that exactly one w remains live after Phase 2 of the operation of P'. Thus, P' determines the value $W(x)$ as claimed. Again, this is in contradiction to the guarantee of our adversary strategy from Claim 3.1.5, so the assumption about P was false. We have proved that $\mathbf{B}(y)$ satisfies property (iii) in the definition of MBFs, and altogether we have shown that $\mathbf{B}(y)$ is a $((1 + \frac{\delta}{2})D_N, \delta)$-MBF, proving Lemma 3.2.6 for $c = 1$ (with $T = (1 + \frac{\delta}{2})D_N, M = D_N^2$). □

3.3 Chapter acknowledgments

I would like to thank Scott Aaronson and Russell Impagliazzo for their support and encouragement, Michael Forbes for valuable proofreading assistance, Brendan Juba and Shubhangi Saraf for helpful discussions, and Suresh Venkatasubramanian for pointing me to the definition of lattice outer measures in [Bir67].

Chapter 4

Limitations of Lower-Bound Methods for the Wire Complexity of Boolean Operators

4.1 Known lower-bound methods for wire complexity

In Section 1.4.3 we reviewed the (rather unsatisfying) state of our knowledge of lower bounds on the number of wires needed to compute Boolean operators, in the arbitrary-gates and $\mathbb{F}_2$-linear gates models. Since there are relatively few lower-bound methods for these models, it is important to understand the power and limitations of existing methods. In this chapter we focus on three such methods.

4.1.1 The Strong Multiscale Entropy method

The first method we study, mentioned earlier in Section 1.4.3, was developed by Cherukhin [Che08a] and used to obtain the best known explicit lower bounds on bounded-depth wire complexity. The bounds apply to the cyclic convolution operator over $\mathbb{F}_2^n$, and are of form $\Omega_d\left(n \cdot \lambda_{d-1}(n)\right)$ for depth $d > 1$.[1] Here, $\lambda_d(n)$ is an unbounded function in n, which grows ever-more-slowly as d increases; its growth is extremely slow even for modest values of d. We have[2]

$$\lambda_1(n) \; = \; \Theta(\sqrt{n}) \; , \quad \lambda_2(n) \; = \; \Theta(\ln n) \; , \quad \lambda_3(n) \; = \; \Theta(\ln \ln n) \; ,$$

and for higher d, $\lambda_d(n) = \lambda^*_{d-2}(n)$. The precise definition is in Section 4.4.2.

The longstanding previous best lower bounds for explicit operators (including cyclic convolution) were of form $\Omega\left(\frac{n \ln^2 n}{\ln \ln n}\right)$ for depth 2 [RTS00] and $\Omega_d\left(\lambda_d(n)\right)$ for $d \geq 3$ [DDPW83, Pud94, AP94], and were based on the *superconcentrator technique* [Val76, Val77]. For depths 2, 3 and for even depths $d \geq 4$, Cherukhin's work gives asymptotic improvements on these older bounds; for odd depths $d \geq 5$, his bounds match the best previous ones from [Pud94]. Cherukhin's lower-bound method does not apply to linear operators. (For $d \geq 3$, the best known lower bounds for computing an explicit linear operator are of form $\Omega_d\left(n \cdot \lambda_d(n)\right)$ [Pud94, p. 215], [GHK$^+$12]. These bounds, along with the $\Omega\left(n\left(\frac{\ln n}{\ln \ln n}\right)^2\right)$ bound for depth 2 from [GHK$^+$12], are valid against circuits with arbitrary gates.)

Cherukhin's method, developed specifically for the convolution operator, was later formulated by Jukna [Juk12, Chap. 13] as a general property of operators that yields a lower bound of form $\Omega_d(n \cdot \lambda_{d-1}(n))$. This operator property is called the *Strong Multiscale Entropy (SME)* property. Very roughly speaking, the SME property states that there is a large "information flow" between many subsets of the input and output coordinates of an operator. The precise definition has two noteworthy aspects. First, the SME property requires for this information flow to be large when measured with respect to many different partitions of the input and output coordinates, at many different "scales" (i.e., varying the size of the input and output blocks). Second, the measure of information flow between an input and output block is defined with respect to a well-chosen set of restrictions of the original operator. The SME property

[1]Cherukhin proved his result for depths 2 and 3 earlier in [Che08b]. The paper [Che08a] contains a unified proof for all constant depths.

[2]The $\lambda_d(\cdot)$ functions are defined differently in [Pud94, GHK$^+$12]. We follow [RS03, Che08a, Juk12] instead, and we have converted the bounds quoted from other papers to match our convention.

will be defined in Section 4.4.2.

The earlier superconcentrator technique works by showing (also using "information flow"-type arguments) that for certain operators F, any circuit to compute F must have a strong connectivity property: it must be a so-called superconcentrator graph. This allows one to apply known lower bounds on the number of edges in bounded-depth superconcentrators (on n input and output vertices). The power of this method is inherently limited, since for $d \geq 3$, the smallest depth-d superconcentrators have $\Theta_d(n \cdot \lambda_d(n))$ edges [DDPW83, Pud94, AP94]. Also, there exist superconcentrators with $O(n)$ wires [Val76, Val77]; such graphs cannot have constant depth, but may have depth that grows extremely slowly in n [DDPW83]. In contrast with the superconcentrator technique, the SME property has an inherently information-theoretic definition, and the associated lower bounds are proved by a combination of graph-theoretic techniques from earlier work [Pud94, RS03] with novel information-theoretic techniques. For constant-depth circuits, no limitations on the method were known prior to our work, and it seemed plausible that the SME property might imply significantly stronger lower bounds by an improved analysis.[3]

4.1.2 Two simpler lower-bound methods

We also study two other lower bound methods, both due to Jukna. These methods are simpler than the SME method, and have only been shown to imply lower bounds for depth 2. However, we feel they are still of interest due to their elegance, and due to the fact that the important depth-2 case is still not well-understood.

The first of these methods is the so-called "entropy method" of Jukna [Juk10a]. Like the SME method, this method is a complexity measure of Boolean operators whose definition is information-theoretic: the method identifies information that passes between certain subsets of inputs and outputs, and argues that there must

[3]For larger depths, some limitations of the SME criterion follow from previous work. In particular, the cyclic convolution operator over $\mathbb{F}_2$, which satisfies the SME property, can be computed in depth polylog(n) using $O(n \log n \log \log n)$ wires. To see this, we first note that cyclic convolution of length n in $\mathbb{F}_2$ easily reduces to multiplying two polynomials in $\mathbb{F}_2[x]$, each of degree at most $2n-1$. For the latter task, we can use an algorithm of Schönhage [Sch77] (see [Pos11]).

be many wires to carry this information. (In fact, the property of operators used by Jukna's entropy method can be viewed as a relaxation of the SME property, as will be apparent from the definitions.) Using this method, Jukna proved bounds of form $\Omega(n^{3/2})$ for the number of wires required in depth-2 circuits for multiplication of two $\sqrt{n}$-by-$\sqrt{n}$ matrices over $\mathbb{F}_2$. Like the SME method, Jukna's entropy method does not yield super-linear lower bounds for computing *linear* operators.

The next lower-bound method we study, also due to Jukna [Juk10b] (building on work of Alon, Karchmer, and Wigderson [AKW90]), does apply to linear operators, and indeed is specific to these operators. Jukna showed that if the columns of a matrix $A \in \mathbb{F}_2^{n \times n}$ have pairwise Hamming distance $\Omega(n)$, then any depth-2 circuit (with arbitrary gates) computing the linear transformation $x \to Ax$ must have $\Omega\left(\frac{n \ln n}{\ln \ln n}\right)$ wires [Juk10b]. This lower-bound criterion applies to a wide range of transformations, including random ones. We will refer to this technique as the "method of pairwise distances."

Jukna's result is actually stronger: the $\Omega\left(\frac{n \ln n}{\ln \ln n}\right)$ lower bound applies to any depth-2 circuit that merely computes Ax correctly when x is a standard basis vector e_i, for $i = 1, \ldots, n$. Such a circuit is said to "represent" the transformation Ax (relative to the standard basis); this is a weaker notion than computing the transformation if we allow non-linear gates. It seems worthwhile to understand how much of the difficulty of computing a linear transformation is "already present" in the simpler task of representing it relative to some basis. In this chapter, we will be broadly interested in the complexity of representing linear transformations relative to various bases; we regard the method of pairwise distances as one particular lower-bound technique within this framework.

4.2 Our contributions

4.2.1 Limitations of entropy-based methods

As our most significant (and most technically involved) result, we show that Cherukhin's lower-bound method, formalized by Jukna as the SME property, is inherently limited as a lower-bound criterion for the wire complexity: there is an explicit operator with the SME property that is computable with $O(n \cdot \lambda_{d-1}(n))$ wires, when $d = 2, 3$, or when $d \geq 6$ is even. For other $d > 1$, this gives an upper bound of $O(n \cdot \lambda_{d-2}(n))$ wires. Thus, the Cherukhin-Jukna analysis of the SME lower-bound criterion is essentially tight.

The operator we exhibit, called the "Dyadic Interval Replication" (DIR) operator, is fairly natural, and can be roughly described as follows. Let $n := 2^k$. The input is a string $x \in \{0,1\}^n$, viewed as a labeling of the leafs of $\mathcal{T}_k$, the complete binary tree of depth k, along with a specified subtree $\mathcal{T}'$ of $\mathcal{T}_k$. The desired output is the labeling $z \in \{0,1\}^n$ in which the leaf labels of $\mathcal{T}'$ in x have been "copied" to all other subtrees of the same height. This operator is designed to create significant information flow between all parts of the input and output; the subtree $\mathcal{T}'$ will be encoded in the input in a way that is chosen to help ensure the SME property.

Our efficient bounded-depth circuits for the DIR operator are built by an induction on the depth d.[4] The basic idea is that, when the subtree $\mathcal{T}'$ to be copied is small, we can "shrink" the input x, discarding most of the labelings outside of $\mathcal{T}'$. We then either perform the replication task in a direct fashion, or, if the input has been shrunk substantially enough, we inductively apply our circuits for lower depths. By carefully optimizing the set of sizes to which we attempt to shrink the input, we obtain the upper bounds quoted above. This approach also shows that the DIR operator has *linear*-sized circuits of depth $d = \alpha(n) + 2$, where $\alpha(n) := \min\{d : \lambda_d(n) \leq 1\}$ is an extremely slowly-growing function. The idea of attacking a problem at different "scales," and applying induction, has appeared earlier in efficient constructions of

[4]Technically, our induction gives circuits to compute a simplified variant, which we then apply to compute the original operator.

bounded-depth superconcentrators [DDPW83] and bounded-depth circuits to compute good error-correcting codes [GHK+12], although the details are different in each case.

We share with earlier authors the belief that, for the cyclic convolution operator, it should be possible to prove significantly better lower bounds for bounded depth—say, bounds of form $\Omega(n^{1+\varepsilon_d})$ for any constant $d > 0$. Our work's message is simply that such lower bounds will have to exploit more of the specific structure of this operator. It seems likely that this will require powerful new ideas. We do hope, however, that our DIR example may be a useful reference point for future work in this area.

Next, we turn to study the limits of Jukna's entropy method. In Section 4.6, we give a simple example of an operator from $2n$ input bits to n output bits, which is computable by depth-3 circuits with $O(n)$ wires but requires $\Omega(n^{3/2})$ wires to compute in depth 2. The operator is a simplified variant of matrix multiplication over $\mathbb{F}_2$, in which one of the two matrices is required to contain exactly one 1-entry. The lower bound follows by the same analysis used in [Juk10a] to prove the same lower bound for ordinary matrix multiplication over $\mathbb{F}_2$. Our example shows that the entropy method as formalized in [Juk10a] does not provide a nontrivial lower-bound criterion for depth-3 circuits.

As super-linear lower bounds are already known for the depth-3 wire complexity of certain operators, our negative result on Jukna's entropy method should be interpreted as a note of caution, rather than as a strong barrier to progress in circuit complexity. However, the operator we define to prove our result is also the first known example of a polynomial separation between depth-2 and depth-3 wire complexities—a finding of independent interest. (A *polylogarithmic* complexity separation between depths 2 and 3 is shown in [GHK+12], for the task of computing the encoding function of certain non-explicit linear codes.)

4.2.2 Results on linear transformations

In the rest of the chapter, we study the complexity of representing linear transformations over $\mathbb{F}_2^n$. While Lupanov [Lup56] showed that random linear transformations require $\Omega(n^2/\ln n)$ wires to compute by linear circuits, Jukna [Juk10b] showed that, if we allow non-linear gates, $O(n \ln n)$ wires suffice to *represent* any linear transformation. (He showed this for the standard basis, but his method extends easily to all other bases.) In Section 4.7, we show that relative to any fixed basis B, most linear transformations require $\Omega(n \ln n)$ wires to represent relative to B. Our result shows that Jukna's upper bound is in general optimal. For our proof, we use a simple trick (similar to a technique in [JS10]) to reduce arbitrary circuits to a special, restricted class; we then apply a standard counting argument.

Recall that Jukna's method of pairwise distances [Juk10b] implies a lower bound of $\Omega\left(\frac{n \ln n}{\ln \ln n}\right)$ on the number of wires needed to represent a large class of linear transformations by depth-2 circuits. Jukna asked whether the "annoying" $(\ln \ln n)^{-1}$ factor in his result could be removed, to match the upper bound he proved for arbitrary matrices. In Section 4.8, we show that in fact it cannot: there is a matrix family $\{A_n \in \mathbb{F}_2^{n \times n}\}$ whose columns have pairwise distance $\Omega(n)$, for which we can compute the transformation $x \to A_n x$ using a depth-2, $\mathbb{F}_2$-linear circuit with $O\left(\frac{n \ln n}{\ln \ln n}\right)$ wires. Our construction involves an application of combinatorial designs defined by polynomials over finite fields.

ln Section 4.9, we show that, for depth-3 circuits, the pairwise-distance method fails completely: there is a matrix family $\{A_n \in \mathbb{F}_2^{n \times n}\}$, whose columns have pairwise distance $\Omega(n)$, and for which we can compute $x \to A_n x$ using a depth-3 linear circuit with $O(n)$ wires. Recently, Gál et al. [GHK+12] proved a related result: there is a linear error-correcting code $L : \{0,1\}^{\Omega(n)} \to \{0,1\}^n$ with minimum distance $\Omega(n)$, whose encoding function is computable by depth-3 linear circuits with $O(n \ln \ln n)$ wires. They also show this is optimal for any such code, even if arbitrary gates are allowed. In fact, they determine fairly precisely the minimal wire complexity of computing a good error-correcting code for all depths $d \geq 2$: for depth 2, the answer is $\Theta\left(n\left(\frac{\ln n}{\ln \ln n}\right)^2\right)$, and for depth $d \geq 3$, the answer is $\Theta_d(n \cdot \lambda_d(n))$. As a corollary, this implies that the pairwise-distance method cannot give bounds better than $\Omega(n \ln \ln n)$ for depth 3; our result sharpens this by removing the $(\ln \ln n)$ factor. Comparing our work with [GHK+12] also shows that, while the generator matrices of good linear codes do have columns with high pairwise distance, the property of being a good code is an inherently stronger lower-bound criterion than the pairwise-distance method.

Finally, in Section 4.10, we show another potential pitfall of circuit-size lower bounds based on hardness of representing linear transformations. We show that for *invertible* linear transformations L, there is always a basis B and a depth-3 circuit C of size $O(n)$ such that C represents L relative to B. (Non-linear gates are provably necessary in this construction.) Thus in attempts to prove new circuit lower bounds for depths greater than 2, we must at least take care in choosing which basis we use to analyze our linear transformation.

4.3 Preliminaries

Throughout the chapter we use $e_1, \ldots, e_n$ to denote the standard basis vectors in $\mathbb{F}_2^n$. We freely identify $\{0, 1\}$ with $\mathbb{F}_2$ when it is convenient. We use $||y||$ to denote the Hamming weight of $y \in \{0, 1\}^n$.

Given a gate g in a circuit C, the *depth* of g is defined as the maximal number of edges (i.e., wires) in any directed path from an input gate to g, where each step in the path follows a wire in C in its direction of information-flow. The depth of C is defined as the maximum depth of any of its gates. When we construct circuits, we will refer to the depth-d gates as being at "Level d." Generally these circuits will not be layered; that is, wires may pass from Level d to any Level $d' > d$.

4.3.1 Wire complexity of operators

A *(total) operator* (or *mapping*) is any function $F : \{0, 1\}^n \to \{0, 1\}^m$. A case of special interest is when $F = L$ is an $\mathbb{F}_2$-*linear* operator; we will also refer to linear operators as *linear transformations.*

A *partial operator* is a function $F : D \to \{0, 1\}^m$, where $D \subseteq \{0, 1\}^n$. For $D' \subseteq D$, let $F|_{D'} : D' \to \{0, 1\}^m$ be the restriction of F to D.

For a total or partial operator F, define $s(F)$ as the minimum number of wires in any circuit (using arbitrary Boolean functions at gates) which computes F. For $d \geq 0$, define $s_d(F)$ as the minimum number of wires in any circuit which computes F and has depth at most d. For linear operators we also study the quantity $s^{\oplus}(L)$, defined as the minimum number of wires in any $\mathbb{F}_2$-linear circuit that computes L. Similarly, define $s^{\oplus}_d(L)$ as the minimum number of wires in a $\mathbb{F}_2$-linear circuit of depth at most d that computes L.

4.3.2 Representing linear operators relative to different bases

Fix a basis B for $\mathbb{F}_2^n$. Say that a linear operator $L : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is *represented relative to* B by the circuit C (with n input and m output gates) if $C(x) = L(x)$ for all $x \in B$. (Definitions in [Juk10b, JS10] applied to the standard basis; we consider more general bases.) Note that if C is a *linear* circuit that represents L relative to some basis B, then in fact C *computes* L.

Let $R_d(L; B)$ be defined as the minimum number of wires in any circuit of depth at most d that represents L relative to B. We let $R(L; B) := \min_{d>0} R_d(L; B)$.

4.3.3 A hashing lemma

The following lemma allows us to "compress" the information in an input string in a wire-efficient way, provided the input is promised to come from a restricted subset. Item 1 of the lemma, which is an especially simple special case, will be used in several sections, while the slightly more technical item 2 will only be used in Section 4.10.

Lemma 4.3.1.

1. *There is a $\mathbb{F}_2$-linear operator $H_{sta} : \mathbb{F}_2^n \to \mathbb{F}_2^{2\lceil\sqrt{n}\rceil}$, computable by a depth-1 circuit with $2n$ wires, and such that for any two distinct standard basis vectors $e_i, e_j \in \mathbb{F}_2^n$, the image vectors $H_{sta}(e_i), H_{sta}(e_j)$ are distinct and each of Hamming weight 2. (We call H_{sta} a "hash mapping" for $\{e_1, \ldots, e_n\}$.)*

2. *Let $D \subset \{0,1\}^n$ be of size n. There is an $\mathbb{F}_2$-linear operator $H : \mathbb{F}_2^n \to \mathbb{F}_2^{\lceil\sqrt{n}\rceil}$, computable by a depth-1 linear circuit with $O(n)$ wires, that satisfies $H(u) \neq$*

$H(v)$ for any two distinct $u, v \in D$.

Proof. **(1.)** For $n \geq 1$, the number of size-2 subsets of $[2\lceil\sqrt{n}\rceil]$ is $\binom{2\lceil\sqrt{n}\rceil}{2} \geq n$. To each $i \in [n]$, we arbitrarily assign a distinct $S_i \subseteq [2\lceil\sqrt{n}\rceil]$ of size 2. Let the input variable x_i be wired to the two gates $h_t, h_{t'}$ where $S_i = \{t, t'\}$, and let each h_t compute the sum mod 2 of its inputs. Then letting $H_{sta}(x) := (h_1(x), \ldots, h_{2\lceil\sqrt{n}\rceil}(x))$, we have

$$H_{sta}(e_i) \;=\; \mathbf{1}_{S_i} \;,$$

where $\mathbf{1}_{S_i} \in \mathbb{F}_2^{2\lceil\sqrt{n}\rceil}$ is the characteristic function of S_i. Our circuit is depth-1, contains $2n$ wires, and computes a mapping with the desired property.

(2.) We will give a construction that works for sufficiently large n; this is enough to prove the statement. Let $h_1, \ldots, h_{\lceil\sqrt{n}\rceil}$ denote the outputs of H. We define H by building the circuit C_H that computes it. Our construction is probabilistic: each input gate is connected to 14 output gates chosen uniformly and independently at random, and each h_t computes the sum (over $\mathbb{F}_2$) of its inputs. (If multiple wires connect the input x_i and output h_t, each wire contributes to the sum. The constant 14 is simply chosen large enough to make the analysis work.) The total number of wires is $14n = O(n)$, as required.

We claim that, with probability $1 - o(1)$ over the randomness in our construction, H is injective on D. To see this, first fix attention to any pair $u, v \in D$ with $u \neq v$. For clarity, reorder the input coordinates so that $u_n \neq v_n$. Condition on any wiring of the outgoing wires from input gates $x_1, \ldots, x_{n-1}$, and consider x_n to have not yet received its assignment of outgoing wires. This incomplete circuit defines a linear transformation $\widetilde{H} : \{0,1\}^n \to \{0,1\}^{\lceil\sqrt{n}\rceil}$ from the inputs to the outputs.

Let $\bar{\mathbf{t}} = (\mathbf{t}(1), \ldots, \mathbf{t}(14)) \in [\lceil\sqrt{n}\rceil]^{14}$ be the 14 uniformly chosen indices of gates to which x_n is to be connected. Assume without loss of generality that $u_n = 0, v_n = 1$. Then $H(u) = \widetilde{H}(u)$, while

$$H(v) \;=\; \widetilde{H}(v) \oplus e_{\mathbf{t}(1)} \oplus \ldots \oplus e_{\mathbf{t}(14)}$$

(here e_t denotes the t-th standard basis vector in $\mathbb{F}_2^{\lceil\sqrt{n}\rceil}$). Let $w := \widetilde{H}(u) \oplus \widetilde{H}(v)$; it follows that

$$H(u) \;=\; H(v) \quad\iff\quad e_{\mathbf{t}(1)} \oplus \ldots \oplus e_{\mathbf{t}(14)} = w \;. \tag{4.1}$$

We will show that, regardless of the value w, the probability $p_w := \Pr[e_{\mathbf{t}(1)} \oplus \ldots \oplus e_{\mathbf{t}(14)} = w]$ satisfies $p_w = o(n^{-2})$. First, $p_w = 0$ if $||w|| > 14$. There are $\binom{\lceil\sqrt{n}\rceil}{k}$ strings of Hamming weight k, and each occurs as $e_{\mathbf{t}(1)} \oplus \ldots \oplus e_{\mathbf{t}}(14)$ with equal probability, so $p_w = O(n^{-3})$ if $||w|| \in [6, 14]$.

Now say $||w|| = k \leq 5$. Given an outcome of $\bar{\mathbf{t}}$ satisfying $e_{\mathbf{t}(1)} \oplus \ldots \oplus e_{\mathbf{t}(14)} = w$, the cancellations which occur imply that we can find $\ell := (14-k)/2$ pairs of indices $\{i_1, j_1, \ldots, i_\ell, j_\ell\} \subseteq [14]$, with no two indices appearing twice, such that

$$\mathbf{t}(i_r) = \mathbf{t}(j_r),\;\; r = 1, 2, \ldots, \ell. \tag{4.2}$$

Each event $[\mathbf{t}(i) = \mathbf{t}(j)]$ occurs with probability $\lceil\sqrt{n}\rceil^{-1}$ if $i \neq j$. The variables $\mathbf{t}(1), \ldots, \mathbf{t}(14)$ are independent, so the probability that Eq. (4.2) holds is $O(n^{-\ell/2}) = O(n^{-9/4})$, using $k \leq 5$.

In each case we find $p_w = o(n^{-2})$, so by Eq. (4.1), we conclude $\Pr[H(u) = H(v)] = o(n^{-2})$. By a union bound over all pairs $u, v \in D$, the probability that H fails to be injective is $\binom{n}{2} \cdot o(n^2) = o(1)$. So our construction of H has the desired property on some setting to the randomness. □

4.4 Entropy and circuit lower bounds

4.4.1 Entropy of operators

Given an operator $F = (f_1, \ldots, f_m) : \{0,1\}^n \to \{0,1\}^m$, define the *entropy*

$$\mathrm{Ent}(F) \;:=\; \log_2 (|\mathrm{range}(F)|)$$

as the logarithm of the number of distinct outputs of F. We have two easy facts, both from [Juk10a]:

Fact 4.4.1. *Suppose we fix some assignment to a subset $I \subseteq [n]$ of the inputs to F, and let $F' : \{0,1\}^{n-|I|} \rightarrow \{0,1\}^m$ be the resulting operator. Then* $\mathrm{Ent}(F') \leq \mathrm{Ent}(F)$.

Fact 4.4.2. *Suppose that there is a subset $S \subseteq [n]$, such that from the value $F(x)$ one can always infer the values of all input bits x_i with $i \in S$. Then,* $\mathrm{Ent}(F) \geq |S|$.

Say we are given an $x \in \{0,1\}^n$, a nonempty set $I \subseteq [n]$, and an $i \in I$. Let $x[I;i]$ denote the vector obtained from x by setting the i^{th} bit to 1, setting the $(i')^{th}$ bit to 0 for each $i' \in I \setminus \{i\}$, and leaving all other bits unchanged.

Letting $F(x)$ be as above, and fixing some output coordinate $j \in [m]$, define the function

$$f_{I,i,j}(x) \ := \ f_j(x[I;i]) \ .$$

Now for $J \subseteq [m]$, define a mapping $F_{I,J} : \{0,1\}^{n-|I|} \rightarrow \{0,1\}^{|I|\cdot|J|}$ by

$$F_{I,J} \ := \ (f_{I,i,j})_{i \in I, j \in J} \ .$$

Note, $F_{I,J}$ has as its domain the bits $\{x_\ell : \ell \notin I\}$. (We will still write $F_{I,J} = F_{I,J}(x)$, however.) We can now state Jukna's entropy-based lower-bound criterion:

Theorem 4.4.3. *[Juk10a] Let $F : \{0,1\}^n \rightarrow \{0,1\}^m$. Let $I_1, \ldots, I_p$ be a partition of $[n]$, and let $J_1, \ldots, J_p$ be a partition of $[m]$ with the same number of parts. Then,*

$$s_2(F) \ \geq \ \sum_{t=1}^{p} \mathrm{Ent}(F_{I_t,J_t}) \ .$$

4.4.2 Strong Multiscale Entropy

Next we define the Strong Multiscale Entropy property, which is a generalization due to Jukna [Juk12, Chap. 13] of a lower-bound method of Cherukhin [Che08a].

For a pair of integers $N, m \geq n_0$, we consider pairs $(\mathcal{I}, \mathcal{J})$ where $\mathcal{I}$ is a collection of subsets of $[N]$ and $\mathcal{J}$ is a collection of subsets of $[m]$. For an integer $p \leq n_0$, we say that $(\mathcal{I}, \mathcal{J})$ form an *n_0-partition at scale p* if:

1. $\mathcal{I}$ consists of p disjoint sets $I_t \subseteq [N]$, with $|I_t| = \lfloor n_0/p \rfloor$;

2. $\mathcal{J}$ consists of $\lfloor n_0/p \rfloor$ disjoint sets $J_{t'} \subseteq [m]$, with $|J_{t'}| = p$.

Say that a family $\{F_N : \{0,1\}^N \to \{0,1\}^m\}_{N>0}$ has the *Strong Multiscale Entropy (SME) property*, if there exists a parameter $n_0 = n_0(N) = \Omega(N)$ along with constants $C, \gamma > 0$ such that, for every N and every $p \in [C\sqrt{n_0}, n_0]$, there exists a pair $(\mathcal{I}, \mathcal{J})$ that form an n_0-partition at scale p, satisfying

$$\mathrm{Ent}(F_{I_t, J_{t'}}) \geq \gamma \cdot n_0 \ , \quad \forall I_t \in \mathcal{I},\ J_{t'} \in \mathcal{J}. \tag{4.3}$$

We also define the *enhanced SME property* similarly to the above, except that we ask for a pair $(\mathcal{I}, \mathcal{J})$ satisfying Eq. (4.3) for all $p \in [C, n_0]$.

To state the lower bounds for operators with the SME property, we need some definitions. We let $g^{(i)}$ denote the i-fold composition of a function $g : \mathbb{Z} \to \mathbb{Z}$. Suppose g satisfies $1 \leq g(n) < n$ for all $n > 1$; we then define $g^* : \{1, 2, 3, \ldots\} \to \{0, 1, 2, \ldots\}$ by

$$g^*(n) \ := \ \min\{i : g^{(i)}(n) \leq 1\} \ .$$

Following conventions in [RS03, Che08a], define a family of slowly-growing functions $\lambda_d(n)$ as follows: let

$$\lambda_1(n) \ := \ \lfloor \sqrt{n} \rfloor, \quad \lambda_2(n) := \lceil \log_2 n \rceil \ ,$$

and for $d > 2$, let

$$\lambda_d(n) \ := \ \lambda^*_{d-2}(n) \ .$$

(Note that $\lambda_3(n) = \Theta(\ln \ln n)$.)

Applying the technique of Cherukhin [Che08a], Jukna proved:

Theorem 4.4.4. *[Juk12, Chap. 13] Suppose the operator family $\{F_N : \{0,1\}^N \to \{0,1\}^m\}$ has the Strong Multiscale Entropy property. Then for any constant $d \geq 2$, any depth-d circuit to compute F_N has $\Omega_d(N \cdot \lambda_{d-1}(N))$ wires.*

4.5 Limitations of the SME lower-bound criterion

In this section we introduce an explicit Boolean operator called the "Dyadic Interval Replication" (DIR) operator, and use it to show that the Strong Multiscale Entropy property does not imply wire complexity lower bounds substantially better than those given by Theorem 4.4.4. We prove:

Theorem 4.5.1. *There is an operator family* $\{\mathrm{DIR}_N : \{0,1\}^N \to \{0,1\}^{\Omega(N)}\}$*, with the enhanced Strong Multiscale Entropy property, for which we have:*

$$s_2\left(\mathrm{DIR}_N\right) \;=\; \Theta(N^{3/2}) \;=\; \Theta\left(N \cdot \lambda_1(n)\right) \;;$$

$$s_3\left(\mathrm{DIR}_N\right) \;=\; \Theta\left(N \ln N\right) = \Theta\left(N \cdot \lambda_2(n)\right) \;;$$

$$s_5\left(\mathrm{DIR}_N\right) \;=\; O\left(N \ln\ln N\right) = O(N \cdot \lambda_3(n)) \;;$$

For even $d = d(N) \geq 6$*,*

$$s_d\left(\mathrm{DIR}_N\right) \;=\; O\left(N \cdot \lambda_{d-2}(N)\right) = O\left(N \cdot \lambda_{d-1}(N)\right) \;,$$

and so for constant, even values $d \geq 6$*,*

$$s_d\left(\mathrm{DIR}_N\right) \;=\; \Theta_d\left(N \cdot \lambda_{d-1}(N)\right) \;.$$

For odd values $d = d(N) \geq 7$*, we have*

$$s_d\left(\mathrm{DIR}_N\right) \;\leq\; s_{d-1}\left(\mathrm{DIR}_N\right) = O(N \cdot \lambda_{d-2}(N)) \;.$$

The lower bounds come from Theorem 4.4.4. In the statements above, we are using the fact that $\lambda_d(N) = \Theta\left(\lambda_{d+1}(N)\right)$ for even values $d = d(N) \geq 4$.

The hidden constants in the $O(\cdot)$ notation above are *independent* of d. Thus, DIR_N is computable by a circuit with $O(N)$ wires, of depth $\alpha(N)+2$, where $\alpha(N) := \min\{d : \lambda_d(N) \leq 1\}$ is an extremely slowly-growing function. On the other hand, the lower bounds from Theorem 4.4.4 hide a multiplicative constant that goes to 0 as $d \to \infty$. So there may be room for some further tightening of the upper or lower bounds for all values of d.

In Theorem 4.5.1, we show that DIR_N satisfies not only the SME property, but also the *enhanced* SME property. We do so to clarify that even this stronger property does not yield significantly better lower bounds than those given by Theorem 4.4.4. We emphasize that our upper bounds for the specific operator DIR_N are also upper limits on the lower bounds that follow in general from the SME property.

4.5.1 The DIR operator

Now we define DIR_N and show it has the SME property. In our work in this section, it will be convenient to index vectors in $\{0,1\}^n$ as $x = (x_0, \ldots, x_{n-1})$, and regard the indices as lying in $\mathbb{Z}_n$. For $a \in \mathbb{Z}_n$, define

$$\mathrm{shift}(x; a) \;:=\; (x_{-a}, x_{1-a}, \ldots, x_{(n-1)-a}) \;,$$

with index arithmetic over $\mathbb{Z}_n$. We also use set addition: for $A, B \subseteq \mathbb{Z}_n$, define $A + B := \{a + b : a \in A, b \in B\}$ (with addition over $\mathbb{Z}_n$). For $i \in \mathbb{Z}_n$, we write $A + i := A + \{i\}$.

We consider input lengths $N = 2 \cdot 2^k + \lceil \log_2 k \rceil$, for $k \geq 1$. We let $n := 2^k$, and we regard inputs of length N to have the form

$$(x, y, r) \in \{0,1\}^{n+n+\lceil \log_2 k \rceil} \;.$$

We will consider r as an integer in $[0, k-1]$.[5] Define the *Dyadic Interval Replication* operator $\mathrm{DIR}_N(x, y, r) : \{0,1\}^N \to \{0,1\}^n$ by the following rule:

1. If $||y|| \neq 1$, output $z := 0^n$.

2. Otherwise, let $i = i(y) \in \mathbb{Z}_n$ be the unique index for which $y_i = 1$. Output the

[5]If k is not a power of 2, some values in $[0, k-1]$ will have more than one encoding; this technicality doesn't affect our arguments.

string z given by

$$z_j \;:=\; \mathrm{shift}(x; i\cdot 2^r)_{(j \bmod 2^r)} \;. \tag{4.4}$$

Let us explain this definition in words. The input vector x divides naturally into $n/2^r = 2^{k-r}$ substrings of length 2^r. The operator DIR_N chooses one of these substrings, and outputs 2^{k-r} consecutive copies of this substring.

We can extend the definition to input lengths $N \geq 6$ not of the above form, by considering the input to be padded with irrelevant bits.

4.5.2 Establishing the SME property for DIR

Lemma 4.5.2. *The family* $\{\mathrm{DIR}_N\}$ *has the enhanced SME property.*

Proof of Lemma 4.5.2. The number of irrelevant bits in the input to DIR_N is not more than twice the number of relevant bits, so for the purposes of our asymptotic analysis, we may assume that N is of form $N = 2\cdot 2^k + \lceil \log_2 k\rceil$ with $k \geq 1$. Let $n := 2^k$, and let $n_0 := n = \Omega(N)$.

Let $p \in [4, n]$ be given. Define collections $\mathcal{I}, \mathcal{J}$ as follows. For $t \in [p]$, let

$$I_t \;:=\; \{0, 1, \ldots, \lfloor n/p \rfloor\} + (t-1)\lfloor n/p\rfloor$$

be the t^{th} consecutive interval of length $\lfloor n/p\rfloor$ in $\mathbb{Z}_n$. For $t' \in [\lfloor n/p \rfloor]$, let

$$J_{t'} \;:=\; \{0, 1, \ldots, p\} + (t'-1)p$$

be the $(t')^{th}$ interval of length p in $\mathbb{Z}_n$. Note that $(\mathcal{I}, \mathcal{J})$ form an n_0-partition at scale p for the input and output lengths of DIR_N.

Say we are given any $t \in [p]$ and $t' \in [\lfloor n/p\rfloor]$; we will show that $\mathrm{Ent}(\mathrm{DIR}_{I_t, J_{t'}}) = \Omega(n) = \Omega(N)$. First, suppose that $p \in [2^\ell, 2^{\ell+1})$, where $\ell > 0$. Then, $J_{t'}$ contains an interval $\widehat{J}$ of form

$$\widehat{J} \;=\; \{0, \ldots, 2^{\ell-1} - 1\} + s\cdot 2^{\ell-1} \;,$$

for some $s \in [0, 2^{k-\ell+1})$. We now fix assignments (y^*, r^*) to part of the input to

$\mathrm{DIR}_{I_t, J_{t'}}$:

$$y^* \ := \ 0^n, \quad r^* \ := \ \ell - 1 \ .$$

Define $\mathrm{DIR}^*_{I_t, J_{t'}}(x) := \mathrm{DIR}_{I_t, J_{t'}}(x, y^*, r^*)$. Using Fact 4.4.1 applied to $\mathrm{DIR}_{I_t, J_{t'}}$, we have $\mathrm{Ent}(\mathrm{DIR}_{I_t, J_{t'}}) \geq \mathrm{Ent}(\mathrm{DIR}^*_{I_t, J_{t'}})$. So it will be enough to lower-bound $\mathrm{Ent}(\mathrm{DIR}^*_{I_t, J_{t'}})$.

Fix any $i \in I_t$. Our assignment $y^* := 0^n$ satisfies

$$||y^*[I_t; i]|| \ = \ 1 \ .$$

Thus for any x, case 2 holds in the definition of $\mathrm{DIR}(x, y^*[I_t; i], r^*)$. Consider any $j \in \widehat{J}$; substituting values into Eq. (4.4), we find

$$\begin{aligned} \left(\mathrm{DIR}_N(x, y^*[I_t; i], r^*)\right)_j \ &= \ \left(\mathrm{shift}(x; i \cdot 2^{\ell-1})\right)_{(j \bmod 2^{\ell-1})} \\ &= \ x_{(j \bmod 2^{\ell-1}) - i2^{\ell-1}} \ . \end{aligned}$$

Thus, from the output of $\mathrm{DIR}^*_{I_t, J_t}(x)$ we can determine x_a, for each $a \in \widehat{J}_{(\bmod 2^{\ell-1})} - 2^{\ell-1} \cdot I_t$. Here, $\widehat{J}_{(\bmod 2^{\ell-1})} := \{j' \in [0, 2^{\ell-1} - 1] : j' = j \mod 2^{\ell-1}$ for some $j \in \widehat{J}\}$. We observe that actually $\widehat{J}_{(\bmod 2^{\ell-1})} = [0, 2^{\ell-1} - 1]$, since $\widehat{J}$ is a consecutive interval of length $2^{\ell-1}$. Fact 4.4.2 now implies that

$$\mathrm{Ent}(\mathrm{DIR}^*_{I_t, J_{t'}}) \ \geq \ \left|[0, 2^{\ell-1} - 1] - 2^{\ell-1} \cdot I_t\right| \ .$$

Recall that I_t is an interval of length $\lfloor n/p \rfloor$. It follows that, with arithmetic taken over the *integers* $\mathbb{Z}$, the set $[0, 2^{\ell-1} - 1] - 2^{\ell-1} \cdot I_t$ is an interval in $\mathbb{Z}$ of size $2^{\ell-1}\lfloor n/p \rfloor$. We conclude that, over $\mathbb{Z}_n$,

$$\begin{aligned} \left|[0, 2^{\ell-1} - 1] - 2^{\ell-1} \cdot I_t\right| &= \min\{n, 2^{\ell-1}\lfloor n/p \rfloor\} \\ &\geq \min\{n, (p/4) \cdot \lfloor n/p \rfloor\} = \Omega(n). \end{aligned}$$

This proves Lemma 4.5.2. □

4.5.3 Efficient bounded-depth circuits for DIR

In this subsection, we prove the upper bounds needed to establish Theorem 4.5.1.

First we prove the upper bound for depth 2, namely $s_2(\mathrm{DIR}_N) = O\left(N^{3/2}\right)$. Our circuit construction will split into two cases, handled separately as follows: first, if $2^r < \sqrt{n}$, the needed substring of x can be copied into $\sqrt{n}$ gates on Level 1 of the circuit, and then copied from this middle level by the output gates. On the other hand, if $2^r \geq \sqrt{n}$, then each output bit can depend on at most $\sqrt{n}$ possible bits of x.

Lemma 4.5.3. $s_2(\mathrm{DIR}_N) = O\left(N^{3/2}\right) = O(N \cdot \lambda_1(N))$.

Proof. As before, we may assume $N = 2 \cdot 2^k + \lceil \log_2 k \rceil$, with $n := 2^k$. For convenience, we will assume further that k is even, so that $\sqrt{n} = 2^{k/2}$ is an integer.

Recall that, when $||y|| = 1$, the output of $\mathrm{DIR}_N(x, y, r)$ will consist of 2^{k-r} consecutive copies of a substring of x of length 2^r. We will design two depth-2 circuits $C^{\downarrow}, C^{\uparrow}$, each with $O\left(N^{3/2}\right)$ wires. $C^{\downarrow}$ will compute DIR_N under the promise that $2^r < \sqrt{n}$; $C^{\uparrow}$ will compute DIR_N provided $2^r \geq \sqrt{n}$. It is then easy to combine these two circuits to get a single circuit computing DIR_N under no assumption. (We apply each of $C^{\downarrow}, C^{\uparrow}$ to the input, merging their corresponding output gates. Each output gate is also wired to the inputs of r, to determine whether it should output the value of $C^{\downarrow}$ or of $C^{\uparrow}$; this takes $O(n \cdot \log_2 k)$ additional wires.)

For $C^{\downarrow}$, the basic idea is that when $2^r < \sqrt{n}$, fewer than $\sqrt{n}$ bits of x actually "matter" for the output; we can extract these bits on Level 1 and distribute them to the appropriate outputs on Level 2. More precisely, we will have $\sqrt{n} + 1$ gates $(s, g_1, \ldots, g_{\sqrt{n}})$ on Level 1 of our circuit $C^{\downarrow}$, each wired to all of (x, y, r). We set $s = 1$ iff $||y|| = 1$. The gates $g_1, \ldots, g_{\sqrt{n}}$ will simply copy the interval of size $2^r < \sqrt{n}$ in x that must be replicated in the output of DIR_N, as determined by x, r, and $i = i(y)$. (This interval of bits from x will be padded with $\sqrt{n} - 2^r$ zeros when copied to Level 1.)

Next, each output bit z_t ($t \in \mathbb{Z}_n$) is wired to all Level 1 gates and to r. We won't give an explicit rule, but it is clear that with these inputs, each z_t can determine its correct output to compute DIR_N (assuming here that $2^r < \sqrt{n}$). The number of wires in $C^{\downarrow}$ is $O\left(n^{3/2} + n(\sqrt{n} + \log_2 k)\right) = O\left(N^{3/2}\right)$, as required.

Now we build $C^{\uparrow}$. The basic idea here is that, assuming $2^r \geq \sqrt{n} = 2^{k/2}$, each output bit z_t depends only on y, r, and on input bits $x_{t'}$ for which $t - t'$ is a multiple of $\sqrt{n}$. Thus, after "compactifying" the relevant information in y into $\sqrt{n}$ bits on Level 1, each output bit can be computed from the Level 1 gates, from r, and from $\sqrt{n}$ bits of x, using $O\left(n^{3/2}\right)$ wires in total. Details follow.

Let $H(y) = H_{sta}(y) = (h_1, \ldots, h_{2\lceil\sqrt{n}\rceil}) : \mathbb{F}_2^n \to \mathbb{F}_2^{2\lceil\sqrt{n}\rceil}$ be the operator from item 1 of Lemma 4.3.1 that is injective on $\{e_1, \ldots, e_n\}$. We implement H on Level 1 of our circuit with $O(n)$ wires, following the construction in Lemma 4.3.1. As in $C^{\downarrow}$, on Level 1 we also include a single gate s, wired to r, that outputs 1 iff $||y|| = 1$. Thus the total number of wires between inputs and Level 1 is $O(n)$, and there are $\sqrt{n} + 1$ gates at Level 1.

Next, each output bit z_t $(t \in \mathbb{Z}_n)$ is wired to all Level 1 gates, to all of r, and to the input bits $(x_t, x_{t+\sqrt{n}}, x_{t+2\sqrt{n}}, \ldots, x_{t+(\sqrt{n}-1)\sqrt{n}})$. Thus our circuit is of depth 2, and the total number of wires to the outputs is $n \cdot ((\sqrt{n} + 1) + \lceil \log_2 k \rceil + \sqrt{n}) = O(n^{3/2})$.

Rather than specifying the output rule for z_t precisely, we argue that this gate has all the information it needs to output $(\mathrm{DIR}_N(x, y, r))_t$ correctly (assuming $2^r \geq \sqrt{n}$). First, if $||y|| \neq 1$, then z_t can learn this and output 0 by looking at s. Otherwise, z_t knows that $||y|| = 1$. In this case, z_t must output the bit $\mathrm{shift}(x; i \cdot 2^r)_{(t \bmod 2^r)} = x_{(t \bmod 2^r) - i2^r}$ (here the outer index arithmetic is over $\mathbb{Z}_n$). This desired bit lies among $(x_t, x_{t+2^r}, \ldots, x_{t+(2^{k-r}-1)2^r})$, and these are contained in the inputs to z_t since 2^r is a multiple of $\sqrt{n}$. Finally, the value $i = i(y)$ can be determined from $H(y)$, because $H(y)$ determines y when $||y|| = 1$. Thus z_t can output the correct value. □

Next, we will develop tools for building more-efficient circuits of higher depths. For depth 3, we will show $s_3(\mathrm{DIR}_N) = O(N \ln N)$. The plan for depth 3 is fairly simple: First, from an input (x, y, r) satisfying $||y|| = 1$, we can extract the index $i = i(y)$ and the value $p := (i \cdot 2^r \bmod n)$ in depth 1, with $n \log_2 n$ wires. Then we show that there is a circuit to compute the appropriate output given (x, i, r, p) using $O(N)$ wires in depth 2, *under the promise* that r equals some fixed value $a \in [0, k-1]$. As there are only $\log_2 n$ possible values of r, we can combine these circuits (merging their output gates) into a single circuit of total depth 3 and with $O(N \ln N)$ wires overall.

To build our circuits for depths 3 and higher, it is useful to introduce some auxiliary operators, which are "easier" versions of DIR_N. The first such operator further restricts the "admissible" values of r to some interval $[a, b] \subseteq [0, k-1]$. Define $\mathrm{DIR}_N^{[a,b]} : \{0,1\}^{2n+\lceil \log_2 k \rceil}$ by

$$\mathrm{DIR}_N^{[a,b]}(x, y, r) := \begin{cases} \mathrm{DIR}_N(x, y, r) & \text{if } r \in [a, b], \\ 0^n & \text{otherwise.} \end{cases}$$

The second simplified operator makes the values i and $p := (i \cdot 2^r \bmod n)$ available in binary. Define $\mathrm{DIR}_N^{bin,[a,b]} : \{0,1\}^{n+k+\lceil \log_2 k \rceil + k}$ by

$$\mathrm{DIR}_N^{bin,[a,b]}(x, i, r, p) := \begin{cases} \mathrm{DIR}_N^{[a,b]}(x, e_i, r) & \text{if } p = i \cdot 2^r \bmod n, \\ 0^n & \text{otherwise.} \end{cases}$$

We are abusing notation slightly, since the input size to $\mathrm{DIR}_N^{bin,[a,b]}$ is actually smaller than $N = 2n + \lceil \log_2 k \rceil$.

The following lemma, which handles a fixed value $r = a$, will be useful.

Lemma 4.5.4. *For any $a \in [0, k-1]$, there is a depth-2 circuit C^a, using $O(n)$ wires, that computes $\mathrm{DIR}_N^{bin,[a,a]}$.*

Proof. Let a be fixed. We include a single gate s on Level 1 that outputs 1 iff all of the following hold:

1. $||y|| = 1$;

2. $p = i \cdot 2^r \bmod n$;

3. $r = a$.

Also on Level 1 of the circuit C^a, we define gates x'_t, for $t \in \{0, 1, \dots, 2^a - 1\}$. Each such gate is wired to the $(k - a)$ most significant bits of p, and to the inputs

$(x_t, x_{t+2^a}, \ldots, x_{t+(2^{k-a}-1)2^a})$. Let $\tilde{p} := p - (p \bmod 2^a)$ be the value obtained by assuming that the unseen bits of p are zero. We then set $x'_t := x_{t-\tilde{p}}$. Note that the needed bit of x falls within the inputs to x'_t. The number of incoming wires to this group of gates is $2^a \cdot \left(2^{k-a} + (k-a)\right) = O(2^k) = O(n)$.

Finally, given an output gate z_j of C^a with $j \in \mathbb{Z}_n$, we set

$$z_j \ := \ x'_{(j \bmod 2^a)} \wedge s \ ,$$

so that the output gates have $2n$ incoming wires in total, and the entire circuit C^a is depth-2 and contains $O(n)$ wires.

We claim that C^a has the desired behavior. To see this, fix any $j \in \mathbb{Z}_n$. First, if $s = 0$ then $z_j = 0$ as needed. Next assume that $s = 1$, so that $\mathrm{DIR}_N^{bin,[a,a]}(x, i, r, p) = \mathrm{DIR}_N(x, e_i, a)$. We compute

$$\begin{aligned} z_j &= x'_{(j \bmod 2^a)} \wedge 1 \\ &= x_{(j \bmod 2^a) - \tilde{p}} \\ &= x_{(j \bmod 2^a) - i2^a} \\ &\text{(since } s = 1 \text{ implies } \tilde{p} = p = i \cdot 2^a \bmod n) \\ &= (\mathrm{shift}(x; i \cdot 2^a))_{(j \bmod 2^a)}, \end{aligned}$$

as needed. This proves the correctness of C^a. □

Lemma 4.5.5. *For any* $0 \le b < k$, $s_2\left(\mathrm{DIR}_N^{bin,[0,b]}\right) = O\left(N \ln N\right)$. *Also,* $s_3\left(\mathrm{DIR}_N\right) = O\left(N \ln N\right) = O(N \cdot \lambda_2(N))$.

Proof. Again assume that $N = 2 \cdot 2^k + \lceil \log_2 k \rceil$, with $n := 2^k$.

First we show $s_2\left(\mathrm{DIR}_N^{bin,[0,k-1]}\right) = O\left(N \ln N\right)$. Let (x, i, r, p) be the inputs. We apply the circuits $C^0, C^1, \ldots, C^b$ from Lemma 4.5.4 to (x, i, r, p). Each such circuit C^a has n outputs, call them $z_{0,a}, \ldots, z_{n-1,a}$. For $t \in \mathbb{Z}_n$, we "collapse" $z_{t,0}, \ldots, z_{t,b}$ into the single output gate z_t (which takes all the inputs of $z_{t,0}, \ldots, z_{t,b}$ as its inputs). This gate is also wired to the input r, and it outputs $z_t := z_{t,r}$.

Let C denote the circuit we have constructed. That C computes $\mathrm{DIR}_N^{bin,[0,b]}$ is immediate. C is of depth 2 since each C^a is of depth 2, and C has $O(N) \cdot (b+1) + n \cdot \lceil \log_2 k \rceil = O(N \ln N)$ wires, since each C^a has $O(N)$ wires and $b < k = \log_2 n$.

Next we show $s_3\left(\mathrm{DIR}_N\right) = O\left(N \ln N\right)$. In our circuit C' for DIR_N, we will assume that the input satisfies $||y|| = 1$. As usual, it is easy to modify this circuit to handle the case where $||y|| \neq 1$.

On Level 1 of our circuit, we compute $i = i(y)$ and $p := i \cdot 2^r \bmod n$. This takes $O(n \ln n)$ wires since i, p are k bits each. Next, we set $b := k-1$ and apply our previously constructed circuit C for $\mathrm{DIR}_N^{bin,[0,k-1]}$ to (x, i, r, p). By definition, the resulting output is $\mathrm{DIR}_N(x, y, r)$. Our construction of C' is of depth $1+2=3$ and contains $O(N \ln N)$ wires. $\square$

To work with depths larger than 3, we will give a technique that allows us to "shrink" the size of an instance of the Dyadic Interval Replication problem, discarding most of the irrelevant bits of x, when the value r is not too large. The next lemma collects two variants of this process.

Lemma 4.5.6. *Let $N = 2 \cdot 2^k + \lceil \log_2 k \rceil$. Let $0 \leq a \leq b \leq k-1$ be given, and let $d = d(N) \geq 1$. Let $N' := 2 \cdot 2^{b-a+1} + \lceil \log_2(b-a+1) \rceil$.*

1. *There is a depth-$(d+2)$ circuit C that computes $\mathrm{DIR}_N^{bin,[a,b]}$; the number of wires in C is*
$$2^{a+1} \cdot s_d\left(\mathrm{DIR}_{N'}^{bin,[0,b-a]}\right) + O\left(N\right) .$$

2. *There is a depth-$(d+3)$ circuit C' that computes $\mathrm{DIR}_N^{[a,b]}$, and has*
$$2^{a+1} \cdot s_d\left(\mathrm{DIR}_{N'}^{bin,[0,b-a]}\right) + O\left(N(k-b)\right)$$
wires.

In each case the $O(\cdot)$ is independent of a, b, d.

Proof. **(1.)** We split into two cases according to whether the input p satisfies $p = 0 \bmod 2^{b+1}$, designing a different depth-$(d+2)$ circuit for each case. It is easy to combine the two circuits using $O(N)$ additional wires. We assume in the following construction that $p \neq 0 \bmod 2^{b+1}$, and then sketch the other, quite similar case; this will double the number of wires, giving the quoted bound.

On Level 1 of our circuit C (for the case $p \neq 0 \bmod 2^{b+1}$), we include gates $x' = (x'_0, \ldots, x'_{2^{b+1}-1})$, where x'_t is wired to $(x_t, x_{t+2^{b+1}}, \ldots, x_{t+(2^{k-b-1}-1)2^{b+1}})$, and also to the $k-b-1$ most significant bits of p, that is, to $p_{b+1}, \ldots, p_{k-1}$. We set

$$x'_t := x_{t-\tilde{p}-2^{b+1}}, \qquad \text{where} \quad \tilde{p} := \textstyle\sum_{\ell=b+1}^{k-1} p_\ell 2^\ell = p - (p \bmod 2^{b+1}).$$

$x_{t-\tilde{p}-2^{b+1}}$ lies among the inputs to x'_t as needed. Computing x' uses $2^{b+1} \cdot (2^{k-b-1} + (k-b-1)) = O(N)$ wires. Also on Level 1 of C, we include a gate s, wired to (i, r, p). We set $s := 1$ iff the following conditions hold: (1) $p = i \cdot 2^r \bmod n$; (2) $r \in [a, b]$. Computing s requires $o(N)$ wires. Define the quantities $i' := i \bmod 2^{b-a+1}$, $r' := \min\{r-a, b-a\}$, $p' := i' \cdot r' \bmod 2^{b-a+1}$, and note that (i', r', p') can all be determined from (i, r, p). On Level 1 of C we also include gates computing (i', r', p'); this takes $O(\ln^2 N) = o(N)$ wires. For $u \in [0, 2^a - 1]$, define $x'(u) = (x'(u)_0, \ldots, x'(u)_{2^{b-a+1}-1})$ by letting

$$x'(u)_\ell := x'_{\ell \cdot 2^a + u} \ .$$

Here we are just introducing new notation that "divides up" x' into the subsequences $x'(0), \ldots, x'(2^a - 1)$.

Next, on Levels 2 through $(d+1)$ of C, for each $u \in [0, 2^a - 1]$ we place a copy of an optimal (wire-minimizing) depth-d circuit computing $\mathrm{DIR}_{N'}^{bin,[0,b-a]}$, to which we provide the values $(x'(u), i', r', p')$ as inputs. Let $z'(u) = (z'(u)_0, z'(u)_1, \ldots, z'(u)_{2^{b-a+1}-1})$ denote the output gates of this circuit.

Finally, for $t \in \mathbb{Z}_n$, we may uniquely write $t = \ell \cdot 2^a + u$, for some $\ell \in [0, 2^{k-a} - 1]$ and $u \in [0, 2^a - 1]$. Then the output gate z_t is defined by

$$z_t \ := \ z'(u)_{\ell \bmod 2^{b-a+1}} \wedge s \ .$$

The total number of wires in our circuit C is $O(N) + 2^a \cdot s_d\left(\mathrm{DIR}_{N'}^{bin,[0,b-a]}\right)$, and the depth of C is $(d+2)$. Next we prove correctness. First, if $s = 0$ then C outputs 0^n

as needed, so assume $s = 1$ (which implies $r' = r - a$). Fix $t \in \mathbb{Z}_{n=2^k}$, and write $t = \ell \cdot 2^a + u$ with ℓ, u as above. We have

$$\begin{aligned}
z_t &= z'(u)_{\ell \bmod 2^{b-a+1}} \wedge s \\
&= \mathrm{shift}(x'(u); i' \cdot 2^{r'})_{(\ell \bmod 2^{r'})} \\
&\text{(using that } 2^{r'} \text{ divides } 2^{b-a+1}) \\
&= x'_{([(\ell \bmod 2^{r'}) - i'2^{r'}] \bmod 2^{b-a+1}) \cdot 2^a + u} \\
&= x'_{([(\ell \bmod 2^{r'}) - i2^{r'}] \bmod 2^{b-a+1}) \cdot 2^a + u} \\
&= x'_{((\ell \cdot 2^a \bmod 2^r) - i2^r) \bmod 2^{b+1} + u} \\
&\text{(using } (c \bmod m) \cdot w = cw \bmod (mw)) \\
&= x'_{2^{b+1} + (\ell \cdot 2^a \bmod 2^r) - (i2^r \bmod 2^{b+1}) + u} \\
&\text{(since } p\text{, a multiple of } 2^r\text{, is } \neq 0 \bmod 2^{b+1}\text{, and } s = 1) \\
&= x_{[2^{b+1} + ((\ell \cdot 2^a + u) \bmod 2^r) - (i2^r \bmod 2^{b+1})] - \tilde{p} - 2^{b+1}} \\
&= x_{(t \bmod 2^r) - (\tilde{p} + (p \bmod 2^{b+1}))} \\
&= x_{(t \bmod 2^r) - p},
\end{aligned}$$

as needed. Finally, the case $p = 0 \bmod 2^{b+1}$ is handled identically except that we let $x'_t := x_{t-\tilde{p}}$. The analysis is very similar. Combining these two circuits adds a factor of 2 to our bound on the wires, which gives the result stated in item 1 above.

(2.) As earlier, we may assume the input to our circuit satisfies $||y|| = 1$, since the other case is easily handled using $O(N)$ additional wires in the circuit. On Level 1 of C', we place gates $p_{k-g}, \ldots, p_{k-1}$, each wired to all the bits of y and to r; these gates output the g most significant bits of $p := (i \cdot 2^r \bmod n)$, where $i = i(y)$ is the unique index for which $y_i = 1$. This takes $g \cdot (n + \lceil \log_2 k \rceil) = O(gN)$ wires. Also on Level 1, we include gates $h_1, \ldots, h_{2\lceil\sqrt{n}\rceil}$ computing the operator $H(y) = H_{sta}(y) : \{0,1\}^n \to \{0,1\}^{2\lceil\sqrt{n}\rceil}$ from Lemma 4.3.1, item 1, that is injective on $\{e_1, \ldots, e_n\}$ and computable in depth 1 with $2n$ wires.

On Level 2 of C', we include gates which compute the quantities (i', r', p') as defined in part 1 of Lemma 4.5.6, relative to $i = i(y)$, r, and $p := i \cdot 2^r \bmod n$. These quantities can be computed with $O(\ln n)$ gates, each wired to r and to $h_1, \ldots, h_{2\lceil\sqrt{n}\rceil}$ (since the value of $H(y)$ determines $i(y)$). This group of gates requires $O(\sqrt{n}\ln n) = o(N)$ input wires overall.

Also on Level 2 of C', we include gates $x'_0, \ldots, x'_{2^{k-g}-1}$, defined just as in part 1 of the Lemma, in terms of x and p. Note that we can compute these values with $O(N)$ wires just as in part 1, since we have computed the g most significant bits of p on Level 1.

Levels 3 through $d+3$ of C' are identical to Levels 2 through $d+2$ of our circuit from part 1 (i.e., the full circuit, combining the two cases we considered). Correctness is proved exactly as before, and the number of gates is $2^{a+1}s_d\left(\mathrm{DIR}_{N'}^{bin,[0,b-a]}\right)+O(gN)$, as required. □

Lemma 4.5.7. $s_5(\mathrm{DIR}_N) = O(N\ln\ln N) = O(N \cdot \lambda_3(N))$.

Proof. As usual we may assume $||y|| = 1$, solving the other case with $O(N)$ additional wires. The idea for our construction is that we will handle the case when $2^r \le n/\log_2 n$ by "shrinking" the input with Lemma 4.5.6, then applying our depth-2 construction from Lemma 4.5.5. We can handle the case $2^r > n/\log_2 n$ by a more straightforward approach since there are only $\approx \log_2\log_2 n$ possible values of r in this range.

For any choice of $b < k$, it follows from the definition of DIR_N that we can write

$$(\mathrm{DIR}_N)_j = \left(\mathrm{DIR}_N^{[0,b]}\right)_j \vee \bigvee_{b<a<k}\left(\mathrm{DIR}_N^{[a,a]}\right)_j, \quad \forall j \in \mathbb{Z}_n \ . \tag{4.5}$$

Set b as the largest value for which $2^b \le n/\log_2 n$. By part 2 of Lemma 4.5.6 with $a := 0$, $\mathrm{DIR}_N^{[0,b]}$ can be computed in depth $5 = 2+3$ with $2\cdot s_2\left(\mathrm{DIR}_{N'}^{bin,[0,b]}\right)+O(N(k-b))$ wires, where $N' = 2\cdot 2^{b+1} + \lceil\log_2(b+1)\rceil$. By Lemma 4.5.5, $s_2\left(\mathrm{DIR}_{N'}^{bin,[0,b]}\right) = O(N'\ln N') = O(2^{b+1}(b+1)) = O((n/\log_2 n)\cdot\log_2 n) = O(n)$. Also, $k - b \le \log_2\log_2 n + O(1)$. Thus the total cost to compute $\mathrm{DIR}_N^{[0,b]}$ in depth 5 is $O(N\ln\ln N)$.

To compute each of $\mathrm{DIR}_N^{[b+1,b+1]}, \ldots, \mathrm{DIR}_N^{[k-1,k-1]}$, we first compute $H(y)$, where $H = H_{sta}$ is the mapping defined within part 1 of Lemma 4.3.1; H is injective on

$\{e_1, \ldots, e_n\}$ and computable in $2n$ wires. On Level 2, we use $(H(y), r)$ to compute $i = i(y)$ and $p := i \cdot 2^r \bmod n$; this takes $O(\sqrt{n} \ln n) = o(n)$ wires. Then we use the depth-2 circuits C^a from Lemma 4.5.4 to compute $\mathrm{DIR}_N^{bin,[a,a]}(x, i, r, p)$ for $a = \{b+1, \ldots, k-1\}$, which give the outputs of $\mathrm{DIR}_N^{[b+1,b+1]}, \ldots, \mathrm{DIR}_N^{[k-1,k-1]}$ we need. Each C^a has $O(n)$ wires, so the total cost of computing $\mathrm{DIR}_N^{[b+1,b+1]}, \ldots, \mathrm{DIR}_N^{[k-1,k-1]}$ is $O(n(k-b)) = O(n \ln \ln n)$.

At Level 5 of our circuit, we combine the outputs of all of our subcircuits: we "merge" the gates giving the values $\left(\mathrm{DIR}_N^{[0,b]}\right)_j, \left(\mathrm{DIR}_N^{[b+1,b+1]}\right)_j, \ldots, \left(\mathrm{DIR}_N^{[k-1,k-1]}\right)_j$ into a single output gate z_j computing the OR of these values. By Eq. (4.5), this circuit computes DIR_N; it is of depth 5 and contains $O(N \ln \ln N)$ wires. This proves the Lemma. □

The next lemma, our key algorithmic tool for depths $d > 5$, gives an inductive construction of ever-more-efficient circuits for $\mathrm{DIR}_N^{bin,[0,k-1]}$ at the cost of increasing the circuit depth.

Lemma 4.5.8. *For even values $d = d(N) \geq 2$, we have $s_d\left(\mathrm{DIR}_N^{bin,[0,k-1]}\right) = O\left(N \cdot \lambda_d(N)\right)$. The $O(\cdot)$ is independent of d.*

Proof. Let $C > 0$ be chosen larger than the implicit constants in the $O(\cdot)$-notation used in all of our previous results, when the bounds are, for convenience, *re-expressed* in terms of the parameter $n = \Theta(N)$; recall that in each case the bound was independent of d and the other parameters. We claim, and prove by induction on even $d \geq 2$, that $s_d\left(\mathrm{DIR}_N^{bin,[0,k-1]}\right) < 40Cn \cdot \lambda_d(n)$. We may assume in what follows that $k > 20$, setting C large enough that the claim is trivially true for $k \leq 20$.

For $d = 2$, Lemma 4.5.5 gives $s_2\left(\mathrm{DIR}_N^{bin,[0,k-1]}\right) < Cn \cdot \lambda_2(n)$, as needed. Now let $d \geq 4$ be even, and consider the statement proved for $d' = d-2$. First, if $\lambda_{d-2}(n) = 1$, the result is trivial; so assume from now on that $\lambda_{d-2}(n) \geq 2$. Define a nondecreasing integer sequence $a_1, a_2, \ldots, a_T$, where

$$a_t \ := \ \lfloor \log_2(n/\lambda_{d-2}^{(t)}(n)) - 20 \rfloor$$

(recalling that $g^{(t)}$ denotes the t-fold composition of g). We let $T := \min\{t : \lambda_{d-2}^{(t)}(n) = 1\}$; thus $T = \lambda_{d-2}^*(n) = \lambda_d(n)$ by the definitions. It is immediate that $\lambda_{d-2}(m) \geq 1$ whenever $m > 1$, so in fact $\lambda_{d-2}^{(T)}(n) = 1$ and all the a_t's are well-defined, with $a_T = k - 20$. Also, $T > 1$ by our assumption $\lambda_{d-2}(n) \geq 2$.

Let $t^* := \min\{t \in [T] : a_t > 0\}$. As $a_T = k - 20$, we can express the interval $[0, k-1]$ as

$$[0, k-1] \;=\; [0, a_{t^*}] \cup [a_{t^*}, a_{t^*+1}] \cup \ldots \cup [a_{T-1}, a_T] \cup [k-19, k-1] \;,$$

and for $j \in \mathbb{Z}_n$ we can write

$$\left(\mathrm{DIR}_N^{bin,[0,k-1]}\right)_j \;=\; \left(\mathrm{DIR}_N^{bin,[0,a_{t^*}]}\right)_j \vee \left(\mathrm{DIR}_N^{bin,[k-19,k-1]}\right)_j \vee \bigvee_{t=t^*+1}^{T} \left(\mathrm{DIR}_N^{bin,[a_{t-1},a_t]}\right)_j \;. \tag{4.6}$$

By the same technique used in Lemma 4.5.7, one can "merge" the outputs of depth-d circuits for the operators $\mathrm{DIR}_N^{bin,[0,a_{t^*}]}$, $\mathrm{DIR}_N^{bin,[a_{t^*+1},a_{t^*+2}]}$, $\ldots$, $\mathrm{DIR}_N^{bin,[a_{T-1},a_T]}$, and $\mathrm{DIR}_N^{bin,[k-19,k-1]}$ to get a depth-d circuit for $\mathrm{DIR}_N^{bin,[0,k-1]}$.

Let $\tilde{n} := 2^{a_{t^*}+1}$, $\tilde{N} := 2 \cdot 2^{a_{t^*}+1} + \lceil \log_2(a_{t^*} + 1) \rceil$. Applying Lemma 4.5.6, part 1 (with $a := 0, b := a_{t^*}$), we find that

$$s_d\left(\mathrm{DIR}_N^{bin,[0,a_{t^*}]}\right) \;\leq\; 2 \cdot s_{d-2}\left(\mathrm{DIR}_{\tilde{N}}^{bin,[0,a_{t^*}]}\right) + Cn \;.$$

If $t^* = 1$, then $2^{a_{t^*}+1} \leq 2^{-19} \cdot (n/\lambda_{d-2}(n))$, and, using the inductive hypothesis,

$$s_{d-2}\left(\mathrm{DIR}_{\tilde{N}}^{bin,[0,a_{t^*}]}\right) \;\leq\; .005C \cdot (n/\lambda_{d-2}(n)) \cdot \lambda_{d-2}(n) \;,$$

so that $s_d(\mathrm{DIR}_N^{bin,[0,a_{t^*}]}) < 1.01Cn$. If $t^* > 1$, then $a_{t^*-1} \leq 0$, so $2^{-a_{t^*-1}} \geq 0$ and

$$\tilde{n} \;=\; 2^{a_{t^*}+1} \;\leq\; 2 \cdot 2^{a_{t^*} - a_{t^*-1}} \;\leq\; [4 \cdot \lambda_{d-2}^{(t^*-1)}(n)/\lambda_{d-2}^{(t^*)}(n)] \;,$$

and

$$\begin{aligned} s_{d-2}\left(\mathrm{DIR}_{\tilde{N}}^{bin,[0,a_{t^*}]}\right) &\leq 40C \cdot [4 \cdot \lambda_{d-2}^{(t^*-1)}(n)/\lambda_{d-2}^{(t^*)}(n)] \cdot 4\lambda_{d-2}(\lambda_{d-2}^{(t^*-1)}(n)) \\ &\leq 640C\lambda_{d-2}^{(t^*-1)}(n) \\ &< Cn \end{aligned}$$

(here using $n = 2^k > 2^{20}$ and $t^* > 1$), so that $s_d(\mathrm{DIR}_N^{bin,[0,a_{t^*}]}) \leq 2Cn$ in this case.

Now consider $t \in [t^*+1, T]$. By Lemma 4.5.6, part 1, we have

$$s_d(\mathrm{DIR}_N^{bin,[a_{t-1},a_t]}) \leq 2^{a_{t-1}+1} \cdot s_{d-2}(\mathrm{DIR}_{N_t}^{bin,[0,a_t-a_{t-1}]}) + Cn ,$$

where

$$N_t := 2 \cdot 2^{a_t-a_{t-1}+1} + \lfloor \log_2(a_t - a_{t-1} + 1) \rfloor .$$

Now $2^{a_t-a_{t-1}+1} \leq [4 \cdot \lambda_{d-2}^{(t-1)}(n)/\lambda_{d-2}^{(t)}(n)]$, so, using the inductive hypothesis, $s_{d-2}(\mathrm{DIR}_{N_t}^{bin,[0,a_t-a_{t-1}]})$ is at most

$$40C \cdot [4 \cdot \lambda_{d-2}^{(t-1)}(n)/\lambda_{d-2}^{(t)}(n)] \cdot (4\lambda_{d-2}(\lambda_{d-2}^{(t-1)}(n)) = 640C\lambda_{d-2}^{(t-1)}(n) .$$

Thus, $s_d(\mathrm{DIR}_N^{bin,[a_{t-1},a_t]})$ is at most

$$2^{a_{t-1}+1}C \cdot (640\lambda_{d-2}^{(t-1)}(n)) + Cn < 1.01Cn ,$$

using the definition of a_{t-1}.

Finally, $\mathrm{DIR}_N^{bin,[k-19,k-1]}$ can be computed with $19Cn$ wires, using 19 applications of Lemma 4.5.4. Combining our cases and applying them to Eq. (4.6), we find that $s_d(\mathrm{DIR}_N^{bin,[0,k-1]})$ is less than $19Cn+2Cn+T\cdot(1.01Cn) < 40Cn\cdot\lambda_d(n)$, since $T = \lambda_d(n)$. This extends the induction to d, completing the proof. $\square$

Lemma 4.5.9. *For even $d \geq 6$, we have $s_d(\mathrm{DIR}_N) = O(N \cdot \lambda_{d-2}(N))$; the $O(\cdot)$ is independent of d.*

Proof. As usual, we may assume the input satisfies $||y|| = 1$ (handling the case

$||y|| \neq 1$ separately with $O(N)$ additional wires).

On Levels 1 and 2 of our circuit C for $\mathrm{DIR}_N(x, y, r)$, we compute $i = i(y)$ and $p := i \cdot 2^r \bmod n$ with $O(N)$ wires, by applying the mapping $H = H_{sta}$ from Lemma 4.3.1, part 1 to y and then applying a brute-force circuit to $(H(y), r)$. Then we apply an optimal depth-$(d-2)$ circuit for $\mathrm{DIR}_N^{bin,[0,k-1]}$ to the tuple (x, i, r, p). This yields the desired output. The number of wires in our circuit is $s_{d-2}(\mathrm{DIR}_N^{bin,[0,k-1]}) + O(N)$, and by Lemma 4.5.8 this is $O(N \cdot \lambda_{d-2}(N))$. □

By collecting the upper bounds for DIR_N in Lemmas 4.5.3, 4.5.5, 4.5.7 and 4.5.9, along with the lower bounds we get from Theorem 4.4.4 and Lemma 4.5.2, we have proved Theorem 4.5.1.

4.6 Limits of Jukna's entropy method, and a separation of depths 2 and 3

In this section, we show:

Theorem 4.6.1. *There is a family of operators $MM' : \{0,1\}^{2n} \to \{0,1\}^n$ for which $s_2(MM') = \Omega(n^{3/2})$ while $s_3(MM') = O(n)$.*

The notation MM' indicates that our operator is a modified (simplified) form of matrix multiplication. The lower bound on MM' for depth 2 will be proved using Jukna's entropy method, Theorem 4.4.3. This example shows that the entropy method cannot be used to prove super-linear wire lower bounds in depth 3.

Proof of Theorem 4.6.1. For any integer $n > 0$, we can find a perfect square $n' = m^2$ in the range $[n, 2n]$. Thus to prove our asymptotic statement, we may assume that $n = m^2$ is itself a perfect square.

We regard the input to MM' as two matrices $X, Y \in \mathbb{F}_2^{m\times m}$. The output is a third matrix $Z \in \mathbb{F}_2^{m \times m}$. Define

$$MM'(X,Y) := \begin{cases} X \cdot Y & \text{if } X \text{ contains exactly one 1-entry,} \\ 0 & \text{otherwise.} \end{cases}$$

Claim 4.6.2. $s_2(MM') \geq m^3 = n^{3/2}$.

Proof. The argument is basically identical to the one used in [Juk10a] to show that multiplying two m-by-m matrices in depth 2 requires m^3 wires. Letting $p := m+1$, we set up and apply Theorem 4.4.3. For $t \in [p-1]$, let I_t be the t-th row of input matrix X, and let J_t be the t-th row of output matrix Z. (Thus $I_p = Y, J_p = \emptyset$, and the p-th part will contribute nothing to the lower bound from Theorem 4.4.3.)

Fix any $t \in [m]$, and values $k, \ell \in [m]^2$. Let $X^{(t,k)}$ be the matrix whose (t,k)-entry is 1 and whose other entries are 0. Note that for any Y,

$$\left(MM'\left(X^{(t,k)}, Y\right)\right)_{t,\ell} \;=\; \left(X^{(t,k)} \cdot Y\right)_{t,\ell} \;=\; Y_{k,\ell}\ .$$

Thus any desired bit of the matrix Y can be recovered from a value in the t-th row of $MM'(X,Y)$ (i.e., in the output block J_t), for some setting to X which has a single 1-entry in the t-th column. Thus Y can be determined from values of $MM'_{I_t,J_t}(0^{m\times m}, Y)$. It follows from Facts 4.4.1 and 4.4.2 that $\text{Ent}(MM'_{I_t,J_t}) \geq m^2$, so by Theorem 4.4.3, $s_2(MM') \geq m \cdot m^2 = m^3$. $\square$

Now we show that $s_3(MM') = O(n)$ by giving a depth-3 circuit C for MM' with $O(n)$ wires.

First, on Level 1 we define $2\lceil\sqrt{n}\rceil = 2m$ "hash gates" $h_1, \ldots, h_{2m}$, which compute the linear transformation $H_{sta}(X) = (h_1(X), \ldots, h_{2m}(X)) : \mathbb{F}_2^n \to \mathbb{F}^{2m}$ given by item 1 of Lemma 4.3.1, applied to the input matrix X. Define $\mathbf{1}_{(i,j)} \in \mathbb{F}_2^{2m}$ as the vector obtained by applying H_{sta} to the input matrix $X^{(i,j)}$ which contains a single 1-entry in its $(i,j)^{th}$ position. By Lemma 4.3.1 the vectors $\mathbf{1}_{(i,j)}$ are pairwise distinct and each of Hamming weight 2.

On Level 1 we also include a "security gate" s. This gate is connected to all the variables in X, and outputs 1 if X has exactly one 1-entry, or 0 otherwise.

Next, on Level 2 we will have a set of "row gates" $r_1, \dots, r_m$, and "column gates" $c_1, \dots, c_m$. The row gate r_k takes $h_1, \dots, h_{2m}$ and s as inputs. We define

$$r_k := \begin{cases} 1 & \text{if } s = 1 \text{ and } (h_1, \dots, h_{2m}) = \mathbf{1}_{(k,j)} \text{ for some } j \in [m], \\ 0 & \text{otherwise.} \end{cases}$$

The column gate c_ℓ takes $h_1, \dots, h_{2m}$, and the ℓ-th column of Y as inputs. We define

$$c_\ell := \begin{cases} Y_{j,\ell} & \text{if } (h_1, \dots, h_{2m}) = \mathbf{1}_{(k,j)} \text{ for some } k \in [m], \\ 0 & \text{otherwise.} \end{cases}$$

Finally, for $k, \ell \in [m]$, on Level 3 we let $Z_{k,\ell}$ be the AND of r_k and c_ℓ.

We argue that C computes MM'. First suppose that X does not have exactly one 1-entry. Then $s = 0$, so all row gates are 0 and $Z = 0^{m \times m}$ as required. Next, suppose X has a single 1-entry in the (i,j) position. Then we have $(h_1, \dots, h_{2m}) = \mathbf{1}_{(i,j)}$, and $s = 1$. It follows that for $k \in [m]$, we have $r_k = [k = i]$. Also, $(c_1, \dots, c_m) = (Y_{j,1}, \dots, Y_{j,m})$. Thus for $\ell \in [m]$, we have $Z_{k,\ell} = [k = i] \wedge Y_{j,\ell}$. This is precisely the (k,ℓ)-entry of $MM'(X,Y)$. Thus C computes MM'.

Finally, we count the wires in C. The subcircuit computing $H_{sta}(X)$ has $O(n)$ gates, by Lemma 4.3.1. The security gate has $m^2 = n$ inputs. Each row and column gate has at most $2m$ inputs, for a total of $\leq (2m)^2$ wires as input to a row or column gate. Each output $Z_{k,\ell}$ has 2 inputs, so the total number of wires is $O(n)$ as desired. This completes the proof of Theorem 4.6.1. $\square$

4.7 Representing random linear operators

In the rest of the chapter, we study the wire complexity of computing and representing linear transformations. (Recall the notion of representing a linear operator L relative to a basis B, and the quantities $R(L;B)$ and $R_d(L;B)$, from Section 4.3.2.)

Jukna [Juk10b] showed:

Theorem 4.7.1. *[Juk10b] Every linear operator $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ can be represented by a depth-2 circuit of $O(n \ln n)$ wires relative to the standard basis.*

An easy modification of his proof shows that for any linear operator $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and any basis B for $\mathbb{F}_2^n$, $R_2(L; B) = O(n \ln n)$. We show that Jukna's upper bound is optimal up to constant factors, by proving the following lower bound on the wire complexity of representing random linear operators:

Theorem 4.7.2. *Fix any basis B for $\mathbb{F}_2^n$. Suppose a random linear operator $L = L_A : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is defined by uniformly selecting its defining matrix $A \in \mathbb{F}_2^{n\times n}$. With probability $1 - o(1)$, we have*

$$R(L; B) \ > \ \frac{n \log_2 n}{5} \ .$$

Theorem 4.7.2 is implied by the following more general result about random *partial* operators (not necessarily linear):

Theorem 4.7.3. *Let $D \subseteq \{0,1\}^n$ be of size $r = r(n) > 1$, and assume $r/\log_2 r \geq \log_2 n$. Then a $(1 - o_n(1))$ fraction of all partial operators $F : D \to \{0,1\}^n$ satisfy*

$$s(F) \ > \ \frac{n \log_2 r}{5} \ .$$

The constant $1/5$ is not optimal, and we do not attempt to optimize it here.

Proof of Theorem 4.7.2. L is distributed as a uniformly random partial operator from B to $\{0,1\}^n$ when we consider its restriction to inputs from a linearly independent set B. Thus the result follows immediately from Theorem 4.7.3. □

Proof of Theorem 4.7.3. We first define an augmented circuit model. Fix a canonical ordering $D = \{x^1, \ldots, x^r\}$ of the possible input strings. In a *free-ID circuit*, the input $x = x^i \in D$ is given along with inputs $z_1, \ldots, z_{\lceil \log_2 r \rceil}$ which give the binary encoding of $i \in [r]$. That is, the circuit is provided with this unique identifier of x "for free," and these bits can be used as inputs to any gate. As before, we can use any function at the circuit gates.

Let $s_{free-ID}(F)$ denote the minimal number of wires in any free-ID circuit which computes F. It is clear that $s_{free-ID}(F) \leq s(F)$, so to prove the Theorem it is enough to prove that $s_{free-ID}(F) > n \log_2 r/5$ holds for a $(1 - o(1))$ fraction of all $F : D \to \{0,1\}^n$.

Suppose $F : D \to \{0,1\}^n$ satisfies $s_{free-ID}(F) \leq n \log_2 r/5$, and let C be an optimal (wire-minimizing) circuit with at most $L := \lfloor n \log_2 r/5 \rfloor$ wires computing F. Besides the input and free-ID gates, all gates with fanin zero are constant (0 or 1), so we may assume C contains at most two such gates. Each wire is input to just one gate, so we can assume the total number of gates of C (inclusive of inputs and free-ID gates) is at most $L + n + \lceil \log_2 r \rceil + 2$. We can then reintroduce useless (fanin-zero) gates as necessary to get exactly this many gates.

Next we make a simple, key observation: optimality of C implies that all gates of C have fanin at most $\lceil \log_2 r \rceil$. To see this, suppose that some gate g of C has fanin greater than $\lceil \log_2 r \rceil$. The value of g on any input $x = x^i$ is determined by i, and hence by the ID variables $z_1, \ldots, z_{\lceil \log_2 r \rceil}$. Thus we can rewire g to have the inputs $z_1, \ldots, z_{\lceil \log_2 r \rceil}$ and output the same result. This modified circuit still computes F but has fewer wires than C, contradicting the minimality of C.

Now we upper-bound the number (call it N_L) of free-ID circuits with at most L wires, exactly $m := L + n + \lceil \log_2 r \rceil + 2$ gates (including the n input gates and $\lceil \log_2 r \rceil$ free-ID gates), and maximum fanin $\lceil \log_2 r \rceil$. By our reasoning above, this will bound the number of operators $F : D \to \{0,1\}^n$ for which $s_{free-ID}(F) \leq n \log_2 r/5$. Our calculations will follow similar ones in [JS10] with minor modifications.

There are at most $(\log_2 r + 2)^m$ sequences of fanins $(d_1, \ldots, d_m)$ we may choose for our gates, where $0 \leq d_i \leq \lceil \log r \rceil$ and $\sum_{i \in [m]} d_i \leq L$. For each such sequence and for $i \in [m]$, we can choose the inputs to the i-th gate in at most $\binom{m}{d_i} \leq m^{d_i}$ ways, and there are at most $2^{2^{d_i}}$ Boolean functions to assign to this gate. Thus,

$$\begin{aligned}
N_L &\le (\log_2 r + 2)^m \prod_{i\in[m]} m^{d_i} \prod_{j\in[m]} 2^{2^{d_j}} \\
&= (\log_2 r + 2)^m \, m^{\sum_{i\in[m]} d_i} 2^{\sum_{j\in[m]} 2^{d_j}} \\
&\le (\log_2 r + 2)^m \, m^L 2^{\sum_{j\in[m]} 2^{d_j}}.
\end{aligned}$$

Taking logs,

$$\begin{aligned}
\log_2(N_L) &\le m\,(\log_2 r + 2) + L\log_2 m + \sum_{j\in[m]} 2^{d_j} \\
&\le 2L\log_2 L + \sum_{j\in[m]} 2^{d_j} + o(L\log_2 L),
\end{aligned}$$

by our settings of m, L.

In a circuit of L wires, counting tells us that fewer than $n/4$ gates have fanin larger than $4L/n$. Since no gate has fanin larger than $\lceil \log_2 r \rceil$, we have

$$\begin{aligned}
\sum_{j\in[m]} 2^{d_j} &\le m2^{4L/n} + (n/4)2^{\lceil \log_2 r\rceil} \\
&\le m2^{4\log_2 r/5} + nr/2 \\
&= nr/2 + Lr^{4/5} + o(Lr^{4/5}).
\end{aligned}$$

Thus,

$$\log_2(N_L) \le 2L\log_2 L + \left(nr/2 + Lr^{4/5}\right) + o\left(L\log_2 L + Lr^{4/5}\right). \tag{4.7}$$

We have $Lr^{4/5} \le nr^{4/5}\log_2 r/5 = o(nr)$. Also,

$$\begin{aligned}
L\log_2 L &\le (1/5)n\log_2 r \log_2(n\log_2 r/5) \\
&< (1/5)n\left[(\log_2 r)(\log_2 n) + \log_2^2 r\right] \\
&\le (1/5)n\left[r + \log_2^2 r\right],
\end{aligned}$$

using our initial assumption $r/\log_2 r \ge \log_2 n$. Plugging into Eq. (4.7), we find

$\log_2(N_L) < nr/2 + 2 \cdot nr/5 + o(nr)$, so for sufficiently large n, $\log_2(N_L) < .95nr$ and $N_L < 2^{.95nr}$.

Finally we compare this to the number of partial operators $F : D \to \{0,1\}^n$. For each of the r inputs in D, there are 2^n possible outputs, so we have $(2^n)^r = 2^{nr}$ many partial operators in total. Thus less than a $2^{-.05nr} = o(1)$ fraction of these satisfy $s_{free-ID}(F) \leq n \log_2 r/5$. $\square$

4.8 Tightness of Jukna's pairwise-distance lower bound for depth 2

Given $A \in \mathbb{F}_2^{n \times m}$, let $\text{Dist}(A) \in \{0, 1 \ldots, n\}$ denote the minimal Hamming distance between any two columns of A. Building on [AKW90], Jukna gave a lower-bound criterion for representing linear operators (proved for the case $n = m$, although a similar result can be given for other cases as well):

Theorem 4.8.1. *[Juk10b] For $A \in \mathbb{F}_2^{n \times n}$, every depth-2 circuit representing the linear transformation $x \to Ax$ relative to the standard basis must have at least $\Omega\left(\text{Dist}(A) \cdot \frac{\ln n}{\ln \ln n}\right)$ wires.*

For random matrices and for some explicit examples, we have $\text{Dist}(A) = \Omega(n)$. In this case, Theorem 4.8.1 gives a lower bound of $\Omega\left(\frac{n \ln n}{\ln \ln n}\right)$, which nearly matches the upper bound from Theorem 4.7.1. It was left open in [Juk10b] whether the $(\ln \ln n)^{-1}$ factor in the bound from Theorem 4.8.1 could be removed, leading to matching upper and lower bounds for a large class of matrices. In this section we show that this cannot be done: there are, in fact, linear transformations with $\text{Dist}(A) = \Omega(n)$, for which the lower bound from Theorem 4.8.1 is tight.

Theorem 4.8.2. *There exists a family of matrices $\{A_n \in \mathbb{F}_2^{n \times n}\}_{n>0}$ for which $\text{Dist}(A_n) = \Omega(n)$, and for which the linear transformation $x \to A_n x$ over $\mathbb{F}_2^n$ can be computed by a depth-2 circuit with $O\left(\frac{n \ln n}{\ln \ln n}\right)$ wires.*

Note that the linear transformations we define can be *computed*, not just represented, using $O\left(\frac{n \ln n}{\ln \ln n}\right)$ wires. Now, Theorem 4.7.2 states that random matrices A

require $\Omega(n \ln n)$ wires even to *represent* by a circuit of any depth. Thus, the difficulty of representing random matrices is not "fully captured" by the property that their columns have pairwise distance $\Omega(n)$.

Our main tool to prove Theorem 4.8.2 is a *combinatorial design*, or set family, defined by low-degree polynomials. Set families of this kind have seen several applications in complexity theory, e.g., in [NW94]. The form we use is given in the following Claim. For a set X, $\mathcal{P}(X)$ denotes the collection of all subsets of X.

Claim 4.8.3. *For any integer $D > 1$, there is an prime $q = O(D)$ and a set family $\mathcal{S} \subset \mathcal{P}(\mathbb{F}_q^2)$, which contains at least D^D sets and satisfies:*

(i) $|S_i| = q$ for all $S_i \in \mathcal{S}$, and for each $a \in \mathbb{F}_q$ we have $|S_i \cap (a \times \mathbb{F}_q)| = 1$;

(ii) $|S_i \cap S_j| \le q/2$ for all $i \ne j$.

Proof. Let q be a prime number in the range $[2D, 4D]$, as guaranteed to exist by Bertrand's postulate. For each nonzero polynomial $P(x) \in \mathbb{F}_q[x]$ of degree at most D, define $S_P \in \mathcal{P}(\mathbb{F}_q^2)$ as the "graph of P,"

$$S_P \ := \ \{(a, b) \in \mathbb{F}_q^2 : P(a) = b\} \ .$$

Let $\mathcal{S}$ contain the sets S_P for each such polynomial. These polynomials are in 1-to-1 correspondence with $\mathbb{F}_q^{D+1} \setminus \{\mathbf{0}\}$, by the mapping which sends a nonzero polynomial to its vector of coefficients. Thus the number of such polynomials is $q^{D+1} - 1 \ge D^D$ and $|\mathcal{S}| \ge D^D$.

It is clear that condition (i) holds, since each S_P is a graph. For condition (ii), note that $(a, b) \in S_P \cap S_Q$ exactly when $P(a) = Q(a) = b$, and distinct polynomials of degree at most D agree on at most $D \le q/2$ values. $\square$

A *sunflower of size k* is a collection of distinct sets $A_1, \ldots, A_k$ (called *petals*), such that the pairwise intersections $A_i \cap A_j$, for $i \ne j$, are all equal to some fixed set C (called the *core*). The lower bound technique of Theorem 4.8.1 works by finding a large sunflower in the incidence pattern of wires in a circuit, and using this sunflower to identify an information bottleneck. The set family $\mathcal{S}$ in Lemma 4.8.3 is a prototypical example of a set family which does not contain sunflowers of too-large size: all sunflowers in $\mathcal{S}$ have size at most q. Thus it is natural to try to use such a set family to show the tightness of Theorem 4.8.1. These were the considerations that led us to the proof of Theorem 4.8.2 given below.

Proof of Theorem 4.8.2. We are going to define the matrix $A_n \in \mathbb{F}_2^{n\times n}$, and its associated transformation L_{A_n}, by defining a depth-2 linear circuit C_n that computes L_{A_n}. Our construction will work for sufficiently large n.

For $y \geq 1$, define $\beta(y) \in \mathbb{R}^+$ as the unique positive solution to $x^x = y$. Direct computation shows that for large n we have $\frac{\ln n}{\ln\ln n} < \beta(n) \leq \frac{(1+o(1))\ln n}{\ln\ln n}$.

Let $\mathcal{S} \subset \mathcal{P}(\mathbb{F}_q^2)$ be the set family given by Lemma 4.8.3, with the setting $D := \lceil \beta(n) \rceil$. We have $|\mathcal{S}| \geq D^D \geq n$ (by definition of $\beta(n)$), so we can assign a distinct set $S_i \in \mathcal{S}$ to each input coordinate $i \in [n]$.

Now we describe the middle level (Level 1) of our depth-2 circuit C_n. Let $q = O(D)$ be the prime number used in defining $\mathcal{S}$. Level 1 of C_n consists of q^2 gates $g_{(a,b)}$ identified with the elements of $\mathbb{F}_q^2$. For $i \in [n]$, the i-th input gate is connected to the middle gate $g_{(a,b)}$ for each $(a,b) \in S_i$. Each $g_{(a,b)}$ is the sum mod 2 of its inputs:

$$g_{(a,b)} := \bigoplus_{i:(a,b)\in S_i} x_i \ .$$

For the output level, we divide the n output bits into q contiguous blocks $B_1, \ldots, B_q$, each of size $|B_a| = \lfloor n/q \rfloor$; the remaining output bits will be identically zero. We re-index the output gates in B_a as

$$B_a = (z_{a,1}, \ldots, z_{a,\lfloor n/q \rfloor}) \ .$$

We fix a collection $V = \{v^0, \ldots, v^{q-1}\} \subseteq \mathbb{F}_2^{\lfloor n/q \rfloor}$, such that any pair of vectors from V disagree on at least a 1/3 fraction of coordinates. This can clearly be achieved for large n, since $\lfloor n/q \rfloor = \omega(q)$. We think of v^a as an "error-correcting encoding" of a.

We determine the output bits as

$$z_{a,\ell} \;:=\; \bigoplus_{0\le b<q} v^b_\ell \cdot g_{a,b} \;=\; \bigoplus_{0\le b<q\colon\, v^b_\ell=1} g_{a,b}\ ,$$

where v^b_ℓ is the ℓ-th bit of v^b. This completes the description of C_n.

Note that each input gate and output gate in C_n is connected to at most $q = O(\beta(n))$ middle gates, so the total number of wires is $O(n \cdot \beta(n)) = O\left(\frac{n \ln n}{\ln \ln n}\right)$, as desired. Also, C_n is $\mathbb{F}_2$-linear as promised, so it defines a matrix $A_n \in \mathbb{F}_2^{n\times n}$ by the relation $C_n(x) \equiv A_n x$. We now argue that $\mathrm{Dist}(A_n) = \Omega(n)$. It is equivalent to show that for each pair $i, j \in [n]$ of distinct indices, $C_n(e_i)$ and $C_n(e_j)$ disagree on $\Omega(n)$ positions.

Fix any $i \in [n]$. For any $a \in \mathbb{F}_q$, condition (i) of Lemma 4.8.3 tells us that the intersection $S_i \cap (a \times \mathbb{F}_q)$ consists of a single element of $\mathbb{F}_q$; call this element $b_i(a)$. We verify that on input vector e_i we have $g_{a,c}(e_i) = 1$ iff $c = b_i(a)$. Thus, for $0 \le \ell < q$ we have $z_{a,\ell} = v^{b_i(a)}_\ell$, so that the restriction of $C_n(e_i)$ to the output block B_a equals $v^{b_i(a)}$.

If $j \in [n] \setminus \{i\}$, then condition (ii) of Lemma 4.8.3 tells us that for at least $q/2$ choices of a we have $b_i(a) \ne b_j(a)$. For such a, the restrictions $C_n(e_i)|_{B_a} = v^{b_i(a)}$, $C_n(e_j)|_{B_a} = v^{b_j(a)}$ disagree on at least $1/3$ of their positions. So the total number of disagreements between $C_n(e_i), C_n(e_j)$ is at least

$$\frac{q}{2} \cdot \frac{\lfloor n/q \rfloor}{3} \;=\; \Omega(n)\ .$$

This shows $\mathrm{Dist}(A_n) = \Omega(n)$, completing the proof. □

By an easy refinement of our argument, for any $\delta > 0$ we can modify the matrices A_n in Theorem 4.8.2 to satisfy $\mathrm{Dist}(A_n) \ge (1/2 - \delta)n$ for sufficiently large n, while the resulting transformation $x \to A_n x$ is still computable in depth 2 with $O_\delta\left(\frac{n \ln n}{\ln \ln n}\right)$ wires. A remaining question is whether we can have $\mathrm{Dist}(A_n) \ge n/2 - o(n)$, with $s_2(A_n) = O\left(\frac{n \ln n}{\ln \ln n}\right)$. To achieve this we would need to change our approach, due to limits on the achievable parameters of combinatorial designs (see [RRV02, Prop. 14]).

It is also natural to wonder whether better lower bounds would be implied by a very strict column-distance condition on $A_n \in \mathbb{F}_2^{n\times n}$, namely $\mathrm{Dist}(A_n) = n/2$. This may be so; however, in [AKW90] it was shown that the *Sylvester* matrices, which satisfy this condition, can be computed using $O(n \cdot \lambda_k(n))$ wires in depth $d = O(k)$.

4.9 The pairwise-distance method fails for depth 3

In this section we show that Jukna's complexity measure $\mathrm{Dist}(A)$ (defined in Section 4.8) does not yield super-linear lower bounds for circuits of depths 3 and higher:

Theorem 4.9.1. *There exists a family of matrices $\{A_n \in \mathbb{F}_2^{n\times n}\}_{n>0}$ for which $\mathrm{Dist}(A_n) \geq n/2 - o(n)$, and such that the linear transformation $x \to A_n x$ over $\mathbb{F}_2^n$ can be computed by an $\mathbb{F}_2$-linear depth-3 circuit with $O(n)$ wires.*

The work of Gál et al. [GHK+12] already implied the existence of a family $\{A_n \in \mathbb{F}_2^{n\times\Omega(n)}\}$ with $\mathrm{Dist}(A_n) = \Omega(n)$, whose associated linear transformations are computable by depth-3 linear circuits with $O(n \ln\ln n)$ wires.

Proof of Theorem 4.9.1. We may assume, by padding if necessary, that the input length n is a perfect square, $n = m^2$. We will define A_n by defining the circuit C_n that computes it. Let $H = H_{sta} : \mathbb{F}_2^n \to \mathbb{F}_2^{2m}$ be the mapping given by item 1 of Lemma 4.3.1, with associated circuit C_H. Let $m' := 2m$. We let the outputs $h_1, \ldots, h_{m'}$ of H occupy Level 1 of C_n, and connect them to the inputs according to C_H. Thus for $e_i \in E$, the gates of C_n's first level compute $H(e_i)$, and the number of wires used for the first level is $O(n)$.

Level 2 will also consist of m' gates, call them $F = (f_1, \ldots, f_{m'})$. Each f_i will be connected to a uniformly random subset of $\{h_1, \ldots, h_{m'}\}$, and will compute the sum over $\mathbb{F}_2$ of its inputs. This requires at most $O(m^2) = O(n)$ wires.

Finally, Level 3 consists of m' blocks of outputs, with each i-th block B_i of size $m/2$. For $i \in [m']$, each gate in B_i simply outputs f_i. Thus Level 3 requires $O(n)$

wires, and the circuit uses $O(n)$ wires in total. Also, C_n is an $\mathbb{F}_2$-linear circuit as promised, since C_H is linear.

We now show that C_n computes a transformation with the desired properties, with probability $1 - o(1)$ over our random choices. The m' gates on Level 2, considered as a linear transformation over the m' Level 1 gates, compute a uniformly random linear transformation from $\mathbb{F}_2^{m'}$ to itself; call this transformation $\widetilde{F}$.

Fix any pair $i, j \in [n]$ with $i \neq j$. Now, as distinct nonzero vectors in $\mathbb{F}_2^{m'}$, the pair $(H(e_i), H(e_j))$ map to uniform, independent images under $\widetilde{F}$. Letting $\Delta(\cdot, \cdot)$ denote Hamming distance, Chernoff-Hoeffding bounds imply that for $\alpha > 0$,

$$\Pr\left[\Delta\left(\widetilde{F}(H(e_i)), \widetilde{F}(H(e_j))\right) < m'/2 - \alpha\sqrt{m' \ln m'}\right] \;=\; \exp\left(-\Omega(\alpha^2 \ln m')\right) \;=\; o(n^{-2})\,,$$

if α is a sufficiently large constant. By a union bound, with probability $1 - o(1)$ we have that $\Delta\left(\widetilde{F}(H(e_i)), \widetilde{F}(H(e_j)\right) \geq m'/2 - O\left(\sqrt{m' \ln m'}\right) = m - o(m)$ for all $i, j \in [n]$, $i \neq j$. Note that by our definition of the output gates of C_n,

$$\Delta(C_n(e_i), C_n(e_j)) \;=\; (m/2) \cdot \Delta\left(\widetilde{F}(H(e_i)), \widetilde{F}(H(e_j)\right)\,,$$

so with high probability, $\Delta(C_n(e_i), C_n(e_j)) \geq n/2 - o(n)$ for all $i, j \in [n]$, $i \neq j$. This proves Theorem 4.9.1. □

4.10 Easy bases for representing linear operators

For a linear transformation L, recall the quantities $R_d(L; B)$ and $R(L; B)$ from Section 4.3. Since computing a transformation is a stronger requirement than representing the transformation, we have

$$s_d(L) \;\geq\; \max_{B:B \text{ a basis for } \mathbb{F}_2^n} R_d(L; B)\,.$$

It is natural to wonder: how close are the left-hand and right-hand sides above? Note that for a random $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we have $s^{\oplus}(L) = \Omega(n^2/\ln n)$, by a standard counting argument. Following Jukna and Schnitger [JS10], we suspect that also $s(L) = \Omega(n^2/\ln n)$ for random L, but this is not known. It was shown by [GHK$^+$12] that $s_2(L) = \Omega\left(n\left(\frac{\ln n}{\ln\ln n}\right)^2\right)$ if $L : \mathbb{F}_2^n \to \mathbb{F}_2^{O(n)}$ is the encoding function for a good linear error-correcting code (and we have explicit examples of these). On the other hand, for any basis B, Jukna's upper bound technique from Theorem 4.7.1 shows that $R_2(L; B) = O(n \ln n)$ for $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$.

So lower bounds for representing a random transformation $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ are probably not even close to optimal bounds for computing L; are provably lower by nearly a $(\ln n)$ factor in some cases; and *never* give bounds of form $\omega(n \ln n)$. However, as mentioned in Section 4.1.1, the largest lower bounds on $s_3(L)$ for an explicit linear transformation L are of form $\Omega(n \cdot \lambda_3(n)) = \Omega(n \ln\ln n)$ [GHK$^+$12], and for higher depths the bounds are weaker still. Thus we feel that the quantities $R_d(L; B)$ are still worth taking seriously as complexity measures. Thinking optimistically, we may ask:

Question 4.10.1. *Given $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$, suppose that $s_d(L) = \Omega(n \ln n)$. Does it follow that $R_d(L; B) = \Omega(n \ln n)$ for some basis B?*

This motivates another, more general question: how do we *find* a good basis B for L, one for which $R_d(L; B)$ is nearly maximized? We don't have an answer to this question. It should also be noted that the lower bound techniques of [Juk10b] which yield Theorem 4.8.1 are specific to the standard basis, so proving lower bounds for representing explicit linear transformations relative to other bases may well be harder.

However, in the present section we will show that, if we consider depth-3 circuits, there are at least some choices for B that definitely *fail* to yield interesting lower bounds. Namely, we will show that, if $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is invertible, then there exists a basis B such that $R_3(L; B) = O(n)$. A uniformly-selected matrix $A \in \mathbb{F}_2^{n \times n}$ is invertible with $\Omega(1)$ probability [CRR90], so this phenomenon applies to many linear transformations. Compare this with Theorem 4.7.2, which tells us that for any *fixed* basis B, $R(L; B) = \Omega(n \ln n)$ for random operators L.

Theorem 4.10.2. *Let $D \subset \{0,1\}^n$ be of size n, and let $G : D \to \{0,1\}^n$ be any partial operator mapping D into the basis vectors $\{e_1, \ldots, e_n\}$. Then G is computable by a depth-3 circuit C with $O(n)$ wires.*

If $L : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is an invertible linear transformation and we set $B := \{L^{-1}(e_1), \ldots, L^{-1}(e_n)\}$, then B is a basis, and it follows from Theorem 4.10.2 that $R_3(L; B) = O(n)$. The circuits used to prove Theorem 4.10.2 involve non-linear gates. This is necessary in general: any linear circuit representing the linear transformation L relative to any basis also computes L, and most linear transformations require $\Theta\left(n^2/\ln n\right)$ wires to compute by a linear circuit [Lup56, Bub86].

Proof of Theorem 4.10.2. Assume, by padding if needed, that n is a perfect square, $n = m^2$. Let H be the hash mapping and C_H the associated depth-1 circuit given by item 2 of Lemma 4.3.1, where in applying Lemma 4.3.1 we let D be the domain of G. We let Level 1 consist of m gates, and use a copy of C_H to connect Levels 0 and 1. Thus, on input $u \in D$, Level 1 computes $H(u)$.

Level 2 of C consists of $2m$ gates, call them $W = (w_1, \ldots, w_{2m})$. Each w_t is wired to every gate on Level 1. To define the behavior of these gates, first choose distinct sets $S_1, \ldots, S_n \subset [2m]$, each of size 2. We have $\binom{2m}{2} \geq n$, so we can do this (we used this idea earlier in the proof of Lemma 4.3.1, item 1). Let $v^i \in \{0,1\}^{2m}$ denote the characteristic vector of S_i.

Recall that $G(u)$ is a standard basis vector for any $u \in D$. Let $i(u)$ be defined by the equation

$$G(u) = e_{i(u)}. \tag{4.8}$$

Define the mapping $W : H(D) \to \{0,1\}^{2m}$ by the rule

$$W(H(u)) := v^{i(u)}. \tag{4.9}$$

This can be done consistently, since H is injective on D. We leave W undefined on other inputs. Recall that each Level 2 gate w_t is wired to see every Level 1 gate, and we have no restrictions on the functions used at gates, so we can indeed implement our choice of W.

Each output (Level 3) gate z_i (for $i \in [n]$) is connected to the two gates $g_t, g_{t'}$ whose indices satisfy $v_t^i = v_{t'}^i = 1$. We let z_i be the AND of g_t and $g_{t'}$.

Consider any input $u \in D$ to our circuit. By Eq. (4.9), the Level 2 gates collectively take on the value $v^{I(u)}$. Thus for $j \in [n]$, we have $z_i = 1$ iff $j = i(u)$. So, by Eq. (4.8) defining $i(u)$, our circuit computes G.

Finally we count the wires. There are $O(n)$ wires between Levels 0 and 1, since C_H has $O(n)$ wires. The number of wires between Levels 1 and 2 is $m \cdot (2m) = O(n)$. Each output gate has two incoming wires, so there are $O(n)$ wires in total. $\square$

4.11 Chapter acknowledgments

I am grateful to Stasys Jukna for many helpful comments.

Chapter 5

New Limits to Classical and Quantum Instance Compression

5.1 Background and new results

5.1.1 Instance compression and parametrized problems

Given an instance of a hard decision problem, we may hope to *compress* that instance into a smaller, equivalent instance, either of the same or of a different decision problem. Here we do not ask to be able to recover the original instance from the smaller instance; we only require that the new instance have the same (yes/no) *answer* as the original. Such *instance compression* may be the first step towards obtaining a solution; this has been a central technique in the theory of *fixed-parameter-tractable* algorithms [DF99, GN07]. Strong compression schemes for certain problems would also have important implications for cryptography [HN10]. Finally, compressing an instance of a difficult problem may also be a worthwhile goal in its own right, since it can make the instance easier to store and communicate [HN10].

It is unknown whether one can efficiently, significantly compress an arbitrary instance of a natural NP-complete language like SAT, the set of satisfiable Boolean formulas.[1] A more limited goal is to design an efficient reduction that achieves com-

[1]If we could efficiently reduce instances of some NP-complete problem to shorter instances of

pression on instances that are particularly "simple" in some respect. To explore this idea, one needs a formal model defining "simple" instances; the versatile framework of *parametrized problems* [DF99] is one such model, and has been extensively used to study instance compression. A parametrized problem is a decision problem in which every instance has an associated *parameter value* k, giving some measure of the complexity of a problem instance.[2] As an example, one can parametrize a Boolean formula ψ by the number of distinct variables appearing in ψ.

An ambitious goal for a parametrized problem P is to compress an arbitrary instance x of the decision problem for P into an equivalent instance x' of a second, "target" decision problem, where the output length $|x'|$ is bounded by a *polynomial* in $k = k(x)$. If P has such a reduction running in time $\mathrm{poly}(|x| + k)$, we say P is *strongly compressible*; we say P is *strongly self-compressible* if the target problem of the reduction is P itself. (In the literature of parametrized problems, a strong self-compression reduction is usually referred to as a *polynomial kernelization.* More generally, a *kernelization* is a polynomial-time self-compression reduction whose output size is bounded by *some* function of the parameter k alone.)

5.1.2 Previous work: results and motivation

Let VAR-SAT denote the Satisfiability problem for Boolean formulas, parametrized by the number of distinct variables in the formula. In their study of instance compression for NP-hard problems, Harnik and Naor [HN10] asked whether VAR-SAT is strongly compressible.[3] They showed that a positive answer would have several significant consequences for cryptography. Notably, they proved that a *deterministic* strong compression reduction for VAR-SAT (with any target problem) would yield a construction of collision-resistant hash functions based on any one-way function—a

the *same* problem, then we could iterate the reduction to solve our problem in polynomial time, implying $\mathsf{P} = \mathsf{NP}$. However, even if $\mathsf{P} \neq \mathsf{NP}$, it is still conceivable that SAT might have an efficient compressive reduction to a different target problem—to the Halting problem, say.

[2]See Section 5.5.1 for details. The parameter k is explicitly given as part of the input to the algorithm.

[3]Strictly speaking, they asked a slightly different question whose equivalence to this one was pointed out in [FS11].

long-sought goal.

In fact, Harnik and Naor showed that for their applications, it would suffice to achieve strong compression for a simpler parametrized problem, the *"OR(SAT) problem:"* this is the Satisfiability problem for Boolean formulas expressed as disjunctions $\psi = \bigvee_{j=1}^{t} \psi_j$, where the parameter is now defined as the maximum bit-length of any sub-formula ψ_j. Strong compression for VAR-SAT easily implies strong compression for OR(SAT). Harnik and Naor defined a hierarchy of decision problems called the "VC hierarchy," which can be modeled as a class of parametrized problems (see [FS11]). They showed that a strong compression reduction for any of the problems "above" OR(SAT) in this hierarchy would also imply strong compression for OR(SAT); this includes parametrized versions of natural problems like the Clique and Dominating Set problems. While Harnik and Naor's primary motivation was to *find* a strong compression scheme for OR(SAT) to use in their cryptographic applications, their work also provides a basis for showing *negative* results: in view of the reductions in [HN10], any evidence against strong compression for OR(SAT) is also evidence against strong compression for a variety of other parametrized problems.

In subsequent, independent work, Bodlaender, Downey, Fellows, and Hermelin [BDFH09] also studied the compressibility of OR(SAT) and of related problems; these authors' motivations came from the theory of *fixed-parameter tractable (FPT)* algorithms [DF99]. An FPT algorithm for a parametrized problem P is an algorithm that solves an arbitrary instance x, with parameter $k = k(x)$, in time $g(k) \cdot \text{poly}(|x| + k)$, for some function $g(\cdot)$. The idea is that even if P is hard in general, an FPT algorithm for P may be practical on instances where the parameter k is small. Now as long as P is decidable, a kernelization reduction for P provides the basis for an FPT algorithm for P: on input x, first compress x, then solve the equivalent, compressed instance. The kernelization approach is one of the most widely-used schemas for developing FPT algorithms.[4] Of course, one hopes to compress by as large an amount as possible, to maximize the efficiency of the resulting FPT algorithm; this motivates the search for

[4]In fact, *every* problem with an FPT algorithm is kernelizable [CCDF97]. This does not mean, however, that the most efficient FPT algorithms always arise from the kernelization approach.

strong self-compression reductions.

Strong self-compression reductions are known for parametrized versions of many natural NP-complete problems, such as the Vertex Cover problem; see, e.g., the survey [GN07]. However, for many other such parametrized problems, including numerous problems known to admit FPT algorithms (such as OR(SAT)), no strong compression reduction is known, to any target problem. Bodlaender et al. [BDFH09] conjectured that no strong self-compression reduction exists for OR(SAT). They made a similar conjecture for the closely-related *"*AND(SAT) *problem,"* in which one is given Boolean formulas $\psi_1, \ldots, \psi_t$ and asked to decide whether $\bigwedge_{j=1}^{t}[\psi_j \in \text{SAT}]$ holds—that is, whether every ψ_j is individually satisfiable. As with OR(SAT), we parametrize AND(SAT) by the maximum bit-length of any ψ_j.

Bodlaender et al. showed that these conjectures (sometimes referred to as the "OR-" and "AND-conjectures") would have considerable explanatory power. First, they showed [BDFH09, Theorem 1] that the nonexistence of strong self-compression reductions for OR(SAT) would rule out strong self-compression for a large number of other natural parametrized problems; these belong to a class we call "OR-expressive problems."[5] Under the assumption that AND(SAT) does not have strong self-compression, Bodlaender et al. ruled out strong self-compression reductions for a second substantial list of problems [BDFH09, Theorem 2], belonging to a class we will call "AND-expressive." Despite the apparent similarity of OR(SAT) and AND(SAT), no equivalence between the compression tasks for these two problems is known.

In light of their results, Bodlaender et al. asked for complexity-theoretic evidence against strong self-compression for OR(SAT) and AND(SAT). Fortnow and Santhanam [FS11] provided the first such evidence: they showed that if OR(SAT) has a strong compression reduction (to any target problem), then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ and the Polynomial Hierarchy collapses to its third level.

The techniques of [BDFH09, FS11] were refined and extended by many researchers to give further evidence against efficient compression for parametrized problems,

[5]See Section 5.5.2. The class of OR-expressive problems is not identical to the class described in [BDFH09], but it is closely related and contains their class, as well as other classes of problems identified in [HN10, BJK11a].

e.g., in [DLS09, DvM10, BTY11, BJK11a, BJK11b, BJK11c, CFM11, HW12, DM12, Kra12]. (See [DM12] for further discussion and references.) As one notable development that is relevant to our work, Dell and Van Melkebeek [DvM10] combined the techniques of [BDFH09, FS11] with new ideas to provide tight compression-size lower bounds for certain problems that *do* admit polynomial kernelizations. Researchers also used ideas from [BDFH09, FS11] in other areas of complexity, giving new evidence of lower bounds for the length of PCPs [FS11, DvM10] and for the density of NP-hard sets [BH08].

Finding evidence against strong compression for AND(SAT) was left as an open question by these works, however. The limits of *probabilistic* compression schemes for OR(SAT) and for OR-expressive problems (including VAR-SAT) also remained unclear. The results and techniques of [FS11] give evidence only against some restrictive sub-classes of probabilistic compression schemes for OR(SAT): schemes with one-sided error, avoiding false negatives; schemes whose error probability is exponentially small in the length of the entire input; and schemes using $O(\log n)$ random bits, where $n = \max_j |\psi_j|$.

5.1.3 Our results

Results on classical compression

We complement the results of [FS11] by providing evidence against strong compression for AND(SAT): we prove that such a compression scheme, to any target problem, would also imply $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. In fact, we show that reductions compressing even by a much more modest amount would imply the same conclusion. For concreteness, we state our most "basic" result on compression of AND(SAT) in a self-contained way below.

Theorem 5.1.1. *Let L be any NP-complete language. Suppose there is a deterministic polynomial-time reduction R that takes an arbitrarily long list of input strings*

$(x^1, \ldots, x^t)$ *and outputs a string* z*, with*

$$z \in L \quad \Longleftrightarrow \quad \bigwedge_{j \in [t]} [x^j \in L] \ .$$

Suppose further that R *obeys the output-size bound* $|z| \leq \left(\max_{j \leq t} |x^j|\right)^{O(1)}$*, with the polynomial bound independent of* t*. Then,* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$*.*

More strongly, we show the following. Suppose there is any second, "target" language L'*, a pair of polynomially-bounded functions* $t(n), t'(n) : \mathbb{N} \to \mathbb{N}$ *with* $t(n) = \omega(1)$ *and* $t'(n) + 1 \leq t(n)/2$*, and a deterministic polynomial-time reduction* $R : \{0,1\}^{t(n) \times n} \to \{0,1\}^{\leq t'(n)}$*, such that*

$$R(x^1, \ldots, x^{t(n)}) \in L' \quad \Longleftrightarrow \quad \bigwedge_{j \in [t(n)]} [x^j \in L] \ .$$

Then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$*.*

We prove Theorem 5.1.1 in Section 5.3. In later sections, we will strengthen and generalize Theorem 5.1.1 using related but more powerful proof techniques. However, we feel it is worthwhile to present a proof of this basic result with a minimum of tools and preliminaries.[6]

The techniques we use to generalize Theorem 5.1.1 will extend naturally (and in a strong fashion) to the *probabilistic* setting with two-sided error, in which we expect the compression reduction to obey some success-probability guarantee on every input. We show (in Theorem 5.7.4, item 1) that any sufficiently "high-quality" compression scheme for AND(SAT) would imply $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Here, "quality" is defined by a certain relationship between the reliability and the compression amount of the reduction, and allows for tradeoff.

We also show (in Theorem 5.7.4, item 2, and Theorem 5.7.5) that beyond a second,

[6]To be precise, in our elementary proof we avoid any overt use of information-theoretic results and concepts; we also avoid the use of the minimax theorem. These tools are central to our stronger and more general approach (which, in particular, is much better suited for analyzing bounded-error reductions), but familiarity with these tools is not necessary to understand Theorem 5.1.1. We mention that the decision to use or avoid information theory in the proof is essentially independent of the choice to use or avoid the minimax theorem.

somewhat more demanding quality threshold, probabilistic compression reductions either for AND(SAT) *or* for OR(SAT) would imply the existence of *non-uniform, statistical zero-knowledge proofs* for NP languages—a stronger (and even more unlikely) consequence than $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. The more-demanding quality threshold in this second set of results is still rather modest, and allows us to prove the following result as a special case:

Theorem 5.1.2 (Informal). *Suppose that either of* AND(SAT) *or* OR(SAT) *is strongly compressible, with success probability* $\geq .5 + 1/\operatorname{poly}(n)$ *for an* AND *or* OR *of length-n formulas. Then there are non-uniform, statistical zero-knowledge proofs for all languages in* NP.

At the other extreme, where we consider compression schemes with more modest compression amounts, but with greater reliability, our techniques yield the following result:

Theorem 5.1.3 (Informal). *Let* $t(n) : \mathbb{N}^+ \to \mathbb{N}^+$ *be any polynomially bounded function. Suppose there is a compression scheme compressing an* AND *of* $t(n)$ *length-n SAT instances into an instance z of a second decision problem L', where* $|z| \leq C \cdot t(n) \log t(n)$ *for some* $C > 0$. *If the scheme's error probability on such inputs is bounded by a sufficiently small inverse-polynomial in n (depending on $t(n)$ and C), then there are non-uniform, statistical zero-knowledge proofs for all languages in* NP. *The corresponding result also holds for* OR*-compression.*[7]

Our results give the first strong evidence of hardness for compression of AND(SAT). They also greatly strengthen the evidence given by Fortnow and Santhanam against *probabilistic* compression for OR(SAT), and provide the first strong evidence against probabilistic compression for the potentially-harder problem VAR-SAT. For *deterministic* (or error-free) compression of OR(SAT), the limits established by our techniques also follow from the techniques of [FS11], which apply given an OR-compression

[7]In fact, *error-free* OR-compression of this sort for SAT would give non-uniform *perfect* zero-knowledge proofs for NP, and error-free AND-compression for SAT would give non-uniform perfect zero-knowledge proofs for coNP; see Theorem 5.7.3.

scheme with compression bound of form $|z| \leq O(t(n) \log t(n))$.[8] On the other hand, we provide somewhat stronger complexity-theoretic evidence for these limits to compression.

Using our results on the infeasibility of compression for AND(SAT) and OR(SAT), and building on [HN10, BDFH09, FS11], we give new complexity-theoretic evidence against strong compressibility for a list of interesting parametrized problems with FPT algorithms. (See Theorem 5.7.7.) This is the first strong evidence against strong compressibility for any of the ten "AND-expressive" problems identified in [BDFH09] (and listed in Section 5.5.2). For the numerous "OR-expressive" problems identified in [HN10, BDFH09] and other works, this strengthens the negative evidence given by [FS11].

Our methods also extend the known results on limits to compression for parametrized problems that do possess polynomial kernelizations: we can partially extend the results of Dell and Van Melkebeek [DvM10] to the case of probabilistic algorithms with two-sided error. For example, for $d > 1$ and any $\varepsilon > 0$, Dell and Van Melkebeek proved that if the Satisfiability problem for N-variable d-CNFs has a polynomial-time compression reduction with output-size bound $O(N^{d-\varepsilon})$, then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Their result applies to co-nondeterministic reductions, and to probabilistic reductions without false negatives; we prove (in Theorem 5.7.11) that the result also holds for probabilistic reductions with two-sided error, as long as the success probability of the reduction is at least $.5 + N^{-\beta}$ for some $\beta = \beta(d, \varepsilon) > 0$. Using reductions described in [DvM10], we also obtain quantitatively-sharp limits to probabilistic compression for several other natural NP-complete problems, including the Vertex Cover and Clique problems on graphs and hypergraphs. (However, the limits we establish do not give lower bounds on the cost of *oracle communication protocols*; these protocols are a generalization of compression reductions, studied in [DvM10], to which that work's results do apply. Trying to extend our results to this model seems like an interesting challenge for further study.)

[8]This is not explicitly shown in [FS11], but follows from the technique of [FS11, Theorem 3.1]; see also [DvM10, Lemma 3] for a more general result that makes the achievable bounds clear.

Our results about AND(SAT) and OR(SAT) follow from more general results about arbitrary languages. For any language L, we follow previous authors and consider the "OR(L) *problem,*" in which one is given a collection $x^1, \ldots, x^t$ of strings, and is asked to determine whether at least one of them is a member of L. We show (in Theorem 5.7.1, item 1) that if a sufficiently "high-quality" probabilistic polynomial-time compression reduction exists for the OR(L) problem, then $L \in \mathsf{NP/poly}$. (As before, "high-quality" is defined by a relation between the reliability of the reduction and the compression amount.) We also show (in Theorem 5.7.1, item 2) that a polynomial-time compression scheme for OR(L) meeting a more demanding standard of quality implies that L possesses non-uniform statistical zero-knowledge proof systems, and lies in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. (For deterministic compression, the conclusion $L \in \mathsf{coNP/poly}$ was established earlier in [FS11].) Applying these results to $L := \overline{\text{SAT}}$ gives our hardness-of-compression results for AND(SAT); applying the second set of results to $L :=$ SAT gives our improved negative results for OR(SAT).

In unpublished work, Buhrman [Buh] constructed an oracle A such that, for every NP^A-complete language L, the decision problem AND(L) does not have a P^A-computable strong compression reduction. This gave earlier, indirect evidence against efficient strong compression for the AND(SAT) problem—or at least, it indicated that exhibiting such a compression reduction would require novel techniques. Now, inspection of the proofs reveals that our new results on compression for OR(L) are all perfectly relativizing. This allows us to identify many more oracles obeying the property of Buhrman's oracle: namely, we may take any A for which $\mathsf{NP}^A \not\subseteq \mathsf{coNP}^A/\mathsf{poly}$. For example, this holds with probability 1 for a *random oracle* [BG81].[9] Such an oracle can also be obtained through a simple diagonalization argument.

For any Boolean function $f : \{0,1\}^* \to \{0,1\}$, we may generalize the OR(L) decision problem to the problem $f \circ L$, in which one is given a collection of strings $x^1, \ldots, x^t$ and must output $f(L(x^1), \ldots, L(x^t))$. So far it would seem that our negative results are fairly specific to the case where the outer "combining function" f

[9]In [BG81] it is shown that $\mathsf{NP}^A \not\subseteq \mathsf{coNP}^A$ for random A; the technique readily extends to give the stronger claim above.

Please note, the main result claimed in this paragraph about (f \circ L) is over-strong and mistaken. This invalidates Sec. 5.7.3 of the thesis, as noted there. We can prove something similar, though; see the revision to "New Limits to Classical and Quantum Instance Compression" on ECCC.

is either AND or OR. However, by an idea suggested in [FS11, Section 7], our negative result on compression for AND(SAT), combined with Fortnow and Santhanam's negative results on compression for OR(SAT), actually implies the following: for *any* Boolean function f that depends on all of its input bits for each input length, no strong compression schemes exist from $f \circ \text{SAT}$ to a target language $L' \in \mathsf{NP}$, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. (See Theorem 5.7.6.) Note the new requirement in this result that L' be in NP; if the combining function f is monotone, this requirement may be dropped. The quantitative bounds we obtain on the achievable compression amount are somewhat weaker for general f than for $f \in \{\text{AND, OR}\}$, however; developing a better understanding of the situation for other combining functions could be another interesting goal for future work.

(End caution)

Results on quantum compression

Up to this point, we have discussed compression reductions in which the input and output are both "classical" bit-strings. However, from the perspective of quantum computing and quantum information [NC00], it is natural to ask about the power of compression reductions that output a *quantum state*. An "n-qubit state" is a quantum superposition over classical n-bit strings; a vast body of research has explored the extent to which information can be succinctly encoded within and retrieved from such quantum states. If quantum computers become a practical reality, quantum instance compression schemes could help to store and transmit hard computational problems; compressing an instance might also be a first step towards its solution by a quantum algorithm.

We propose the following quantum generalization of classical instance compression: a *quantum compression reduction* for a language L is a quantum algorithm that, on input x, outputs a quantum state ρ on some number q of qubits—hopefully with $q \ll |x|$, to achieve significant compression. Our correctness requirement is that there should exist *some* quantum measurement $\mathcal{M}_q$, depending only on q, such that for every x compressing to q qubits, $\mathcal{M}_q(\rho) = L(x)$ holds with high probability over the inherent randomness in the measurement $\mathcal{M}_q(\rho)$. We do not require that $\mathcal{M}_q$ be an efficiently-performable measurement; this is by analogy to the general version of the classical compression task, in which the target language of the reduction may be arbitrarily complex.

Our results for quantum compression are closely analogous to our results in the classical case. First, we show that for any language L, if a sufficiently "high-quality" quantum polynomial-time compression reduction exists for the OR(L) problem, then L possesses a *non-uniform, 2-message quantum interactive proof system* (with a single prover). Second, we show that a sufficiently higher-quality quantum polynomial-time compression reduction for OR(L) implies that L possesses a *non-uniform quantum statistical zero-knowledge proof system.* Remarkably, the two "quality thresholds" in our quantum results are essentially *the same* as in the corresponding results for the classical case.[10] It follows that, unless there exist surprisingly powerful quantum proofs of unsatisfiability for Boolean formulas, the limits we establish for probabilistic compression of AND(SAT) and OR(SAT) hold just as strongly for quantum compression.[11]

5.1.4 Our techniques

In this section we will focus on describing our strongest and most general techniques. As mentioned earlier, we also present a similar, but more "elementary" approach to prove Theorem 5.1.1. We will give some self-contained intuition about that approach in Section 5.3. That strategy bears some similarities to work of Fortnow and Santhanam [FS11] on the hardness of compression for OR(SAT). In particular, it shares an *incremental* approach to defining non-uniform advice for a proof system; in each case, the stage-based construction makes progress in correctly classifying more and more strings of a given input length.

[10]We do place a minor additional restriction on quantum compression reductions for OR(L): we require that the reduction, on input $(x^1, \ldots, x^t)$, outputs a quantum state of size determined by $(\max_j |x^j|)$ and t.

[11]We remark that *3-message* quantum interactive proofs are known to be fully as powerful as quantum interactive proofs in which polynomially many messages are exchanged [Wat03], and that these proof systems are equal in power to PSPACE in the uniform setting [JJUW11]. However, 2-message quantum proof systems seem much weaker, and are not known to contain coNP.

The overall approach

We first describe our techniques for the classical case; these form the basis for the quantum case as well. Our first two general results, giving complexity upper bounds on any language L for which $\mathrm{OR}(L)$ has a sufficiently high-quality compression reduction (Theorem 5.7.1, items 1 and 2), are both based on a single reduction that we describe next. This reduction applies to compression reductions mapping some number $t(n) \le \mathrm{poly}(n)$ of inputs of length n to an output string z of length $|z| = O(t(n) \log t(n))$.

Fix any language L such that $\mathrm{OR}(L)$ has a possibly-probabilistic compression reduction

$$R(x^1, \ldots, x^t) : \; \{0,1\}^{t \times n} \;\longrightarrow\; \{0,1\}^{\le t'} \;,$$

with some target language L', along with parameters t', t satisfying $t' \le O(t \log t) \le \mathrm{poly}(n)$.[12] We will use R to derive upper bounds on the complexity of L. (The reader may keep in mind the main intended setting $L = \overline{\mathrm{SAT}}$, which we will use to derive our hardness results for the compression of AND(SAT). No special properties of this language will be used in the argument, however.)

A simple, motivating observation is that if we take a string $y \in L$ and "insert" it into a tuple $\overline{x} = (x^1, \ldots, x^t)$ of elements of $\overline{L}$, replacing some x^j to yield a modified tuple $\overline{x}'$, then the values

$$R(\overline{x}) \;,\; R(\overline{x}')$$

are *different* with high probability—for, by the "OR-respecting" property of R, we will with high probability have $R(\overline{x}) \in \overline{L'}$, $R(\overline{x}') \in L'$. More generally, for any *distribution* $\mathcal{D}$ over t-tuples of inputs from $\overline{L}$, let $\mathcal{D}[y, j]$ denote the distribution obtained by sampling $\overline{x} \sim \mathcal{D}$ and replacing x^j with y; then the two output distributions

$$R(\mathcal{D}) \;,\; R(\mathcal{D}[y, j])$$

are *far apart* in statistical distance. (Of course, the strength of the statistical-distance

[12]Here we pay exclusive attention to R's behavior on tuples of strings of some equal length n.

lower bound we get will depend on the reliability of our compression scheme.)

We want this property to serve as the basis for an interactive proof system by which a computationally powerful Prover can convince a skeptical polynomial-time (but non-uniform) Verifier that a string y lies in L. The idea for our initial, randomized protocol (which we will later derandomize) is that Prover will make his case by demonstrating his ability to *distinguish* between the two R-output distributions described above, when Verifier privately chooses one of the two distributions, samples from it, and sends the sample to Prover.[13] But then to make our proof system meaningful, Verifier also needs to *fool* a cheating Prover in the case $y \notin L$. To do this, we want to choose $\mathcal{D}, j$ in such a way that the distributions $R(\mathcal{D}), R(\mathcal{D}[y, j])$ are as close as possible whenever $y \notin L$.

We may not be able to achieve this for an index j that is poorly-chosen. For instance, $R(\overline{x})$ may always copy the first component x^1 as part of the output string z, so taking $j = 1$ would fail badly. To get around this, we choose our replacement index j *uniformly at random*, aiming in this way to make R "insensitive" to the insertion of y.[14] As R is a compression scheme, it doesn't have room in its output string to replicate its entire input, so there is reason for hope.

This invites us to search for a distribution $\mathcal{D}^*$ over $\left(\overline{L}_n\right)^t$ with the following properties:

(i) For every $y \in \overline{L}_n$, if we select $j \in [t]$ uniformly then the expected statistical distance $\mathbb{E}_j\left[||R(\mathcal{D}^*) - R(\mathcal{D}^*[y, j])||_{\text{stat}}\right]$ is "not too large;"[15]

(ii) $\mathcal{D}^*$ is efficiently sampleable, given non-uniform advice of length poly(n).

Condition (i) is quite demanding: we need a single distribution $\mathcal{D}^*$ rendering R insensitive to the insertion of *any* string $y \in \overline{L}_n$—a set which may be of exponential

[13]Interactive proofs based on distinguishing tasks have seen many uses in theoretical computer science, and indeed we will rely upon known protocols of this kind in our work; see Section 5.4.4.

[14]We emphasize that the "insensitivity" we are looking for is *statistical*; we are not asking that y have small effect on the output of R for most *particular* outcomes to $\overline{x} \sim \mathcal{D}$. This latter goal may not be achievable, e.g., if R outputs the sum of all its input strings x^i taken as vectors over $\mathbb{F}_2^n$.

[15]For our purposes, it actually suffices to bound $||R(\mathcal{D}^*) - R(\mathcal{D}^*[y, \mathbf{j}])||_{\text{stat}}$, where $\mathbf{j}$ is a uniform value sampled "internally" as part of the distribution. In our streamlined proof of Theorem 5.1.1, we will use this idea. However, our techniques will yield the stronger property in condition (i) above, and this is the course we will follow in proving our general results.

size. Condition (ii) is also a strong restriction: $\overline{L}_n$ may be a complicated set, and in general we can only hope to sample from distributions over $\left(\overline{L}_n\right)^t$ in which t-tuples are formed out of a fixed "stockpile" of $\mathrm{poly}(n)$ elements of $\overline{L}_n$, hard-coded into the non-uniform advice.

Remarkably, it turns out that such a distribution $\mathcal{D}^*$ can always be found. In fact, in item (i), we can force the two distributions to be non-neglibly close (with expected statistical distance $\leq 1 - \frac{1}{\mathrm{poly}(n)}$) whenever the output-size bound t' obeyed by R is $O(t \log t)$; the distributions will be much closer when $t' \ll t$. We call our key technical result (Lemma 5.6.6), guaranteeing the existence of such a $\mathcal{D}^*$, the *"Disguising-Distribution Lemma."*

Assuming this lemma for the moment, we use $\mathcal{D}^*$ as above to reduce any membership claim for L to a distinguishing task for a Prover-Verifier protocol. Given any input y, we've constructed two distributions $\mathfrak{R} = R(\mathcal{D}^*)$ and $\mathfrak{R}' = R(\mathcal{D}^*[y, \mathbf{j}])$ (with $\mathbf{j}$ uniform), where each distribution is sampleable in non-uniform polynomial time. Our analysis guarantees some lower bound $D = D(n)$ on $||\mathfrak{R} - \mathfrak{R}'||_{\mathrm{stat}}$ in the case $y \in L$, and some upper bound $d = d(N)$ on this distance when $y \notin L$. (These parameters depend on the reliability and compression guarantees of R.) If $D(n) - d(n) \geq \frac{1}{\mathrm{poly}(n)}$, we can give non-uniform distinguishing protocols for L, which can converted to public-coin protocols and then non-uniformly derandomized to show that $L \in \mathsf{NP/poly}$. Also, if $D(n)^2 - d(n) \geq \frac{1}{\mathrm{poly}(n)}$ then, using a powerful result due to Sahai and Vadhan [SV03], we can derive a non-uniform, statistical zero-knowledge proof system for L. This also implies $L \in \mathsf{NP/poly} \cap \mathsf{coNP/poly}$.

The Disguising-Distribution Lemma

The Disguising-Distribution Lemma, informally described in Section 5.1.4, is a statement about the behavior of $R(x^1, \ldots, x^t)$ on a specified product subset S^t of inputs ($S = \overline{L}_n$ in our application). This lemma is a "generic" result about the behavior of compressive mappings; it uses no properties of R other than R's output-size bound.[16]

[16]Indeed, in our application we have essentially no control on R's behavior when we consider its restriction to inputs from S^t, so a generic result is needed.

In view of its generality and interest, we are hopeful that the lemma will find other applications.

Our proof of this lemma uses two central ideas. First, we interpret the search for the "disguising distribution" $\mathcal{D}^*$ as a two-player game between a "disguising player" (choosing $\mathcal{D}^*$) and an opponent who chooses y; we can then apply simple yet powerful principles of game theory. Second, to build a winning strategy for the disguising player, we will exploit an information bottleneck in R stemming from its compressive property.[17]

To describe the proof, it is helpful to first understand how one may obtain the distribution $\mathcal{D}^*$ if we drop the efficient-sampleability requirement on $\mathcal{D}^*$, and focus on the "disguising" requirement (condition (i)). To build $\mathcal{D}^*$ in this relaxed setting, we will appeal to the *minimax theorem* for two-player, zero-sum games; applied here, it tells us that to guarantee the existence of a $\mathcal{D}^*$ that succeeds in disguising all strings $y \in \overline{L}_n$, it is enough to show how to build a $\mathcal{D}^*_Y$ that succeeds in expectation, when y is sampled from some fixed (but arbitrary) distribution Y over $\overline{L}_n$.

Here, a natural idea springs to mind: let $\mathcal{D}^*_Y$ just be a product distribution over t copies of Y! In this case, inserting $y \sim Y$ into $\mathcal{D}^*_Y$ at a random location is equivalent to conditioning on the outcome of a randomly-chosen coordinate of a sample from $\mathcal{D}^*_Y$. The intuition here is that, due to the output-size bound on R, the distribution $R(\mathcal{D}^*_Y)$ shouldn't have enough "degrees of freedom" to be affected much by this conditioning.

We show (in Lemma 5.10.4, with similar, alternative results presented in Sections 5.9 and 5.10) that for any product distribution $\overline{x} \sim (\mathcal{D}_1, \ldots \mathcal{D}_t)$ over t-tuple inputs to R, conditioning on the value of $x^j \sim \mathcal{D}_j$ for a uniformly-chosen index $j \in [t]$ has bounded expected effect on the output distribution $R(\overline{x})$. That is, the *expected statistical distance* between the pre- and post-conditioned distributions is bounded non-negligibly away from 1 (provided that $t' \leq O(t \log t)$). We refer to this important property of R as *"distributional stability."*

In our original proof that our compressive mapping R is distributionally stable,

[17]This is hardly the first work in which such a bottleneck plays a crucial and somewhat unexpected role. For example, an interesting and slightly similar application of information-theoretic tools to the study of *metric embeddings* was found recently by Regev [Reg11].

we gave a simple (non-constructive) way to use R as a one-shot *encoding method* for independent, unbiased bits $b_1, \ldots, b_t$. The encoding Enc has a desirable property: for each component $j \in [t]$ whose expected "influence" on the output distribution of R is noticeable (when we fix a single value $x^j \sim \mathcal{D}_j$), our encoding transmits b_j with noticeable advantage over a random guess. We can then deduce strong upper bounds on the influence of a typical component j, using the output-size bound on R and elementary information-theoretic bounds on the reliability of compressive encodings. This analysis succeeds when $t' \leq t - 2$. In our original draft, we used more elaborate techniques (which involved modifying the mapping R itself) to analyze the case when $t \leq t' \leq O(t \log t)$.

Several researchers pointed out to us that the distributional stability property can be established in a different way, using Kullback-Leibler divergence and an inequality due to Pinsker (see Theorem 5.4.7). This approach allows us to analyze the case when $t \leq t' \leq (1 + \varepsilon)t$, for a modest $\varepsilon > 0$. As this author noted later, the divergence-based approach can be combined with an alternative to Pinsker's inequality—a bound due to Vajda (Theorem 5.4.8; see [FHT03, RW09] for more information on both of these inequalities)—to show that the mapping R has a non-negligible amount of distributional stability as long as $t' \leq O(t \log t)$. Thus we feel that the divergence-based approach is ultimately the most convenient one to work with in general; this is the approach we now use in the main body of the chapter.

Colleagues additionally helped us to understand that the distributional stability property for mappings with $t' \leq (1 + \varepsilon)t$ can also be established using other similar, known results that follow from the same divergence/Pinsker-based techniques: a lemma of Raz [Raz98], and the "Average Encoding Theorem" of Klauck et al. [KNTSZ07]. The latter was used in [KNTSZ07] to identify a stability property for trace and Hellinger distance metrics, for the inputs to a problem in quantum communication complexity; this was used for a round-elimination argument. Their proof is for inputs drawn from the uniform distribution, but extends readily to general distributions and can be used to derive the kind of lemma we need. We describe these alternative proofs of distributional stability in Section 5.9,[18] and we describe our own original, encoding/decoding-based approach in Section 5.10. We feel that all of these approaches to proving distributional stability are interesting and worth understanding.

Using the distributional-stability property of compressive mappings under product input-distributions, we then establish a certain "sparsified variant" of this property (Lemma 5.6.3), which allows us to replace each $\mathcal{D}_j$ with a small set sampled from $\mathcal{D}_j$;[19] this is an important tool in addressing the efficient-sampleability requirement on our desired $\mathcal{D}^*$. Using this variant, we use the minimax theorem to show (in Lemma 5.6.4) that there exists a *distribution* $\mathfrak{D}$ over product input-distributions to R—with each product distribution defined over small subsets of S—such that, in expectation, $\mathfrak{D}$ disguises the random insertion of any string $y \in S$ at a uniformly-chosen position j. Finally, in Lemma 5.6.6 we obtain our desired "disguising distribution" $\mathcal{D}^*$ as a sparsified version of $\mathfrak{D}$, using a result due to Lipton and Young [LY94] and, independently, to Althöfer [Alt94], that guarantees the existence of sparsely-supported, nearly-optimal strategies in 2-player, zero-sum games.

Extension to the quantum case

Our techniques for studying quantum compression are closely analogous to the classical case. The main technical difference is that the output $R(\mathcal{D})$ of our compression reduction, on any input distribution $\mathcal{D}$, is now a (mixed) *quantum state.* In this setting, to carry out an analogue of the argument sketched in Sections 5.1.4 and 5.1.4 and fool a cheating Prover, we need a "disguising distribution" for R that meets a modified version of condition (i) from Section 5.1.4:

(i') For every $y \in \overline{L}_n$, if we select $j \in [t]$ uniformly then, *for any quantum measurement* $\mathcal{M}$, the expected statistical distance $\mathbb{E}_j\left[||\mathcal{M}(R(\mathcal{D}^*)) - \mathcal{M}(R(\mathcal{D}^*[y, j]))||_{\text{stat}}\right]$

[18]Russell Impagliazzo suggested the use of Raz's lemma; Salil Vadhan also helped me to understand the connection. Ashwin Nayak and S. Vadhan suggested direct proofs of distributional stability based on divergence and Pinsker's inequality, which we now use as our main approach. Dieter van Melkebeek also suggested the relevance of Pinsker's inequality. I thank all of these researchers.

[19]For convenience in the proof, we assume $\mathcal{D}_j = \mathcal{D}_{j'}$ for all j, j'.

is not too large.

A basic measure of distance between quantum states, the *trace distance*, is relevant here: if two states ρ, ρ' are at trace distance $||\rho - \rho'||_{\mathrm{tr}} \le \delta$, then for any measurement $\mathcal{M}$, the statistical distance $||\mathcal{M}(\rho) - \mathcal{M}(\rho')||_{\mathrm{stat}}$ is at most δ. (In fact, this property *characterizes* the trace distance.) Thus to satisfy condition (i'), it will be enough to construct $\mathcal{D}^*$ so as to upper-bound $\mathbb{E}_j[||R(\mathcal{D}^*) - R(\mathcal{D}^*[y,j])||_{\mathrm{tr}}]$, for uniformly-chosen j. We do this by essentially the same techniques as in the classical case. The one significant difference is that here, we need to establish a "stability property" for trace distance, analogous to the stability property for statistical distance described in Section 5.1.4. This can be obtained using the same basic divergence-based techniques as in the classical case, with the help of suitable tools from quantum information theory.[20]

5.1.5 Organization of the chapter

In Section 5.2, we present the "bare minimum" of preliminaries needed to understand our proof of Theorem 5.1.1. We present this proof in Section 5.3.

The rest of the chapter is devoted to proving stronger and more general results. In Section 5.4, we give the additional needed preliminary material for our work, including our definitions of compression reductions. In Section 5.5, we formally introduce parametrized problems and AND- and OR-expressive problems. In Section 5.6, we prove the main technical lemmas we use to obtain our results on limits of efficient instance compression (with alternative proofs of the first such lemma appearing in Sections 5.9 and 5.10). Our results for the classical setting are proved in Section 5.7, and the quantum results are proved in Section 5.8. Finally, in Section 5.12 we present questions for future study.

[20]Our original approach to proving distributional stability also admits a quantum version, although we no longer present it here.

5.2 Preliminaries I

Definition 5.2.1. *The* binary entropy function $H(\alpha) : [0,1] \to [0,1]$ *is defined by*

$$H(\alpha) \; := \; -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$$

on $(0,1)$*, with* $H(0) = H(1) := 0$*.*

$\binom{n}{k}$ denotes the binomial coefficient $n!k!/(n-k)!$. We will use the following standard, simple bound (see, e.g., [vL99, Chapter 1]) on the number of binary strings of low Hamming weight:

Fact 5.2.2. *For* $t \in \mathbb{N}$ *and* $\alpha \in (0, .5)$*, we have*

$$\sum_{0 \le \ell \le \alpha t} \binom{t}{\ell} \; \le \; 2^{H(\alpha)t} \; .$$

5.2.1 Statistical distance and distinguishability

All distributions in this chapter will take finitely many values; let $\text{supp}(\mathcal{D})$ be the set of values assumed by $\mathcal{D}$ with nonzero probability, and let $\mathcal{D}(u) := \Pr[\mathcal{D} = u]$.

For a probability distribution $\mathcal{D}$ and $t \ge 1$, we let $\mathcal{D}^{\otimes t}$ denote a t-tuple of outputs sampled independently from $\mathcal{D}$. We let $\mathcal{U}_K$ denote the uniform distribution over a multiset K.

The *statistical distance* of two distributions $\mathcal{D}, \mathcal{D}'$ over a shared universe of outcomes is defined as

$$||\mathcal{D} - \mathcal{D}'||_{\text{stat}} \; := \; \frac{1}{2} \sum_{u \in \text{supp}(\mathcal{D}) \cup \text{supp}(\mathcal{D}')} |\mathcal{D}(u) - \mathcal{D}'(u)| \; .$$

We will use the following familiar "distinguishability interpretation" of the statistical distance. Suppose a value $b \in \{0,1\}$ is selected uniformly, unknown to us, and a sample $u \in U$ is drawn from $\mathcal{D}$ if $b = 0$, or from $\mathcal{D}'$ if $b = 1$. We observe u, and our goal is to correctly guess b. It is a basic fact that, for *any* $\mathcal{D}, \mathcal{D}'$, our maximum achievable success probability in this "distinguishing" experiment is precisely $\frac{1}{2}(1 + ||\mathcal{D} - \mathcal{D}'||_{\text{stat}})$. Furthermore, the optimal distinguishing algorithm may without loss of generality be a deterministic "maximum-likelihood" rule $ML(b|u)$: guess "$b = 1$" if and only if $\Pr[b = 1|u] \geq 1/2$. Similarly, we define a maximum-likelihood rule $ML(X|Y)$ for guessing any random variable X based on the observed value of any other random variable Y: simply guess the likeliest value of X conditioned on the observation (breaking ties arbitrarily).

The following fact follows from the distinguishability characterization of $||\cdot||_{\text{stat}}$; it is a convenient weakening of that principle.

Fact 5.2.3. *If X, Y are random variables over some shared domain S, and $\Delta := ||X - Y||_{\text{stat}}$, then there exists a subset $T \subseteq S$ such that*

$$\Pr_{x\sim X}[x \in T] \;\geq\; \Delta \quad \textit{and} \quad \Pr_{y\sim Y}[y \notin T] \;\geq\; \Delta \;.$$

We will also use the following facts:

Fact 5.2.4. *If X, Y are random variables over some shared domain S, and $R(X)$ is any (possibly randomized) function taking inputs from S, then*

$$||R(X) - R(Y)||_{\text{stat}} \;\leq\; ||X - Y||_{\text{stat}} \;.$$

Fact 5.2.5 ([SV03], Fact 2.3)**.** *Suppose (X_1, X_2, Y_1, Y_2) are distributions on a shared probability space Ω, that X_1 is independent of X_2, and that Y_1 is independent of Y_2. Then,*

$$||(X_1, X_2) - (Y_1, Y_2)||_{\text{stat}} \;\leq\; ||X_1 - Y_1||_{\text{stat}} + ||X_2 - Y_2||_{\text{stat}} \;.$$

5.3 Proof of Theorem 5.1.1

This section presents a proof that the "AND-conjecture" of Bodlaender, Downey, Fellows, and Hermelin [BDFH09] holds true unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. As discussed earlier, in this section we aim for a proof that avoids information theory and the minimax theorem. In later sections we will obtain stronger and more general results with these tools.

It will be convenient to consider mappings $R : \{0,1\}^{t\times n} \to \{0,1\}^{\le t'}$, for fixed n, t, t'. For $A \subseteq \{0,1\}^n$, let $\mathbf{R}_A$ denote the distribution $\mathbf{R}_A := R(\mathcal{U}_A^{\otimes t})$, and for each each $a \in \{0,1\}^n$, define the distribution

$$\mathbf{R}_A[a] := R(\mathcal{U}_A^{\otimes(\mathbf{j}-1)}, a, \mathcal{U}_A^{\otimes(t-\mathbf{j})}) \ ,$$

where $\mathbf{j} \sim \mathcal{U}_{[t]}$.

Define the *standout factor*

$$\beta(a, A) \ := \ ||\mathbf{R}_A[a] - \mathbf{R}_A||_{\text{stat}} \ . \tag{5.1}$$

The basic idea of our proof of Theorem 5.1.1 is as follows: we will show that for each $n > 0$, there exists a poly(n)-size collection of poly(n)-size sets $A_i \subseteq L_n$,[21] such that every other element $x \in L_n$ will have standout factor $\beta(x, A_i) < 1 - \Omega(1)$ for at least one A_i. On the other hand, each $x \notin L_n$ will have standout factor 1 against each A_i.[22] Thus, if a polynomial-time Verifier "quizzes" a Prover by randomly sampling, either from $\mathbf{R}_{A_i}$ or from $\mathbf{R}_{A_i}[x]$ on each i, then Prover will be able to reliably guess which distribution was sampled from if and only if $x \notin L$. By known results, this leads to the conclusion $L \in \mathsf{coNP/poly}$.

Toward this end, the next lemma is our main technical tool:

Lemma 5.3.1. *Let $R : \{0,1\}^{t\times n} \to \{0,1\}^{\le t'}$ be given. Let $A \subseteq \{0,1\}^n$ be a set of size $M \ge 100t$, and suppose that we select $a^* \sim \mathcal{U}_A$. Then if t is sufficiently large and $t' < 2(t-1)$, we have*

$$\mathbb{E}\left[\beta(a^*, A \setminus a^*)\right] \ \le \ 1 - 10^{-4} \ . \tag{5.2}$$

Lemma 5.3.1 establishes that certain distributions are (at least slightly) "sta-

[21](here, $L_n = L \cap \{0,1\}^n$)

[22]We note that this amounts to a weakened version of the Disguising-Distribution Lemma of Section 5.6.

ble" under modification. Related facts, with information-theoretic proofs, appear in [Raz98, KNTSZ07] (see Section 5.9), and these can be readily used to obtain our lemma. A distinctive aspect of Lemma 5.3.1, however, is that it establishes the closeness of the output distribution of R induced by an input to R containing a string a^*, to one from an input distribution to R that does not support a^*. This "apples-to-oranges" comparison is key to our application of Lemma 5.3.1: we will use it to build small (poly(n)-size) subsets of L_n that serve as helpful non-uniform advice to prevent *exponential*-size chunks of L_n from being accepted by Verifier. In the "minimax-free" proof being presented here, we will do so in an iterative fashion until all of L_n is "covered" by our advice. This is reminiscent of the incremental approach of Fortnow and Santhanam [FS11] to defining their advice, in their proof that the OR-conjecture holds unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

In the more general proofs we give in later sections, Lemmas 5.6.2 and 5.6.3 will play a role analogous (but not identical) to that of Lemma 5.3.1 in the current proof.

Proof of Lemma 5.3.1. Suppose to the contrary that $\mathbb{E}\left[\beta(a^*, A \setminus a^*)\right] > 1 - 10^{-4}$. Call $a \in A$ "distinctive" if $\beta(a, A \setminus a) \geq .99$; the measure β is bounded by 1, so more than a .99 fraction of $a \in A$ are distinctive.

For each $a \in A$, let $T = T_a$ be the set given by Fact 5.2.3, with $X := \mathbf{R}_{A\setminus a}[a], Y := \mathbf{R}_{A\setminus a}$; then for all distinctive $a \in A$, we have

$$\Pr_{z \sim \mathbf{R}_{A\setminus a}[a]}[z \in T_a] \ \geq \ .99 \ , \qquad \Pr_{z \sim \mathbf{R}_{A\setminus a}}[z \notin T_a] \ \geq \ .99 \ . \tag{5.3}$$

Let us index A as $A = \{a^1, \ldots, a^M\}$. Define a random R-input $\mathbf{x} = (x^1, \ldots, x^t) \sim \mathcal{U}_A^{\otimes t}$, and for $i \in [M]$ let $\mathrm{Incl}_i(\mathbf{x})$ be the indicator variable for the event that at least one of the elements x^j is equal to a^i. We also define the indicator variable

$$\mathrm{Corr}_i(\mathbf{x}) \ := \ [\mathrm{Incl}_i(\mathbf{x}) \Leftrightarrow (R(\mathbf{x}) \in T_{a^i})] \ = \ \neg[\mathrm{Incl}_i(\mathbf{x}) \oplus (R(\mathbf{x}) \in T_{a^i})] \ .$$

The idea is that $R(\mathbf{x}) \in T_{a^i}$ "suggests" that a^i was included among the inputs to R, while $R(\mathbf{x}) \notin T_{a^i}$ suggests the opposite; $\mathrm{Corr}_i(\mathbf{x})$ checks whether the suggestion given is correct.

It is easy to see that, if we condition on $[\mathrm{Incl}_i(\mathbf{x}) = 0]$, then $R(\mathbf{x})$ is distributed as $\mathbf{R}_{A\setminus a^i}$. In this case, the conditional probability that $[\mathrm{Corr}_i(\mathbf{x}) = 1]$ holds is at least .99, provided a^i is distinctive.

On the other hand, suppose we condition on $[\mathrm{Incl}_i(\mathbf{x}) = 1]$. Then the conditional probability that a^i appears *twice* among the coordinates of $\mathbf{x}$ is, by basic counting, at most $t/M \leq .01$. (After conditioning on any leftmost occurrence of a^i, there are at most $t-1$ indices which could contain the next occurrence of a^i; and each plays this role with probability at most $1/M$.) Thus under this conditioning, $R(\mathbf{x})$ is .01-close to the distribution $\mathbf{R}_{A\setminus a^i}[a^i]$, so that $[\mathrm{Corr}_i(\mathbf{x}) = 1]$ holds with probability at least $.99 - .01 = .98$ if a^i is distinctive.

It is also the case that $\sum_{i\in[M]} \mathrm{Incl}_i(\mathbf{x}) \geq .95t$ with probability at least .99 (for sufficiently large t), since $t/M \leq .01$. Combining all of our work, we find that for large enough t, with probability at least .5 the following conditions hold:

1. $\sum_{i\in[M]} \mathrm{Incl}_i(\mathbf{x}) \geq .95t$;
2. $\sum_{i\in[M]} [\mathrm{Incl}_i(\mathbf{x}) \wedge \mathrm{Corr}_i(\mathbf{x})] \geq .9t$;
3. $\sum_{i\in[M]} \mathrm{Corr}_i(\mathbf{x}) \geq .9M$.

Say that $\mathbf{x}$ is *good* if all of these conditions hold.

Now fix any R-output $z \in \{0,1\}^{\leq t'}$; we are going to derive an upper bound U on the number of good inputs $\mathbf{x}$ for which $R(\mathbf{x}) = z$. Since every $\mathbf{x}$ maps to a string of length $\leq t'$ under R, it will follow that

$$2^{t'+1} \geq \frac{.5|A^{\times t}|}{U} = \frac{.5M^t}{U}, \tag{5.4}$$

which will yield a contradiction to our settings.

First, suppose $z \in T_{a^i}$ for more than $t + .1M$ indices $i \in [M]$. Then for any $\mathbf{x}$ such that $R(\mathbf{x}) = z$, there are more than $.1M$ indices for which $\mathrm{Incl}_i(\mathbf{x}) = 0$ yet $z \in T_{a_i}$. For such i, $\mathrm{Corr}_i(\mathbf{x}) = 0$. Thus $\mathbf{x}$ is not good. So to have *any* good inputs $\mathbf{x}$ map to it under R, z must satisfy

$$|\{i : z \in T_{a^i}\}| \;\le\; t + .1M \; . \tag{5.5}$$

Next, suppose $R(\mathbf{x}) = z$ and that $\mathbf{x} = (x^1, \ldots, x^t)$ contains more than $.15t$ components x^j whose value is any element $x^j = a^i \in A$ for which $z \notin T_{a^i}$. If $\mathbf{x}$ is good, then by property 1 of good inputs, among these components we can find a subcollection of more than $.1t$ components x^j whose values are pairwise distinct. For each $a^i = x^j$ in this subcollection, we have $\mathrm{Incl}_i(\mathbf{x}) = 1$ yet $\mathrm{Corr}_i(\mathbf{x}) = 0$. Thus $\sum_{i \in [M]} [\mathrm{Incl}_i(\mathbf{x}) \wedge \mathrm{Corr}_i(\mathbf{x})] < .9t$, so $\mathbf{x}$ is not good—a contradiction. Thus any good $\mathbf{x}$ for which $R(\mathbf{x}) = z$ can contain at most $.15t$ components x^j whose value $x^j = a^i$ satisfies $z \notin T_{a^i}$.

Combining this observation with Eq. (5.5), there is a set $A' \subseteq A$ (depending on z) of size at most $t + .1M \le .11M$, such that for any good $\mathbf{x}$ mapping to z under R, at least $.85t$ components x^j satisfy $x^j \in A'$. We can now bound the number of good inputs $\mathbf{x}$ mapping to z under R; any such $\mathbf{x}$ is specifiable by:

- a set of at most $.15t$ “exceptional” indices $j \in [t]$;
- the values of x^j on these exceptional indices;
- the values of x^j on all other indices, which must lie in A'.

The number of such $\mathbf{x}$ is at most

$$\begin{aligned}
\sum_{0 \le t' \le .15t} \binom{t}{t'} M^{t'} (.11M)^{t-t'} &\le (.11)^{.85t} M^t \cdot \sum_{0 \le t' \le .15t} \binom{t}{t'} \\
&\le\; (.11)^{.85t} M^t \cdot 2^{H(.15)t} \\
&<\; 4^{-t} M^t \; ,
\end{aligned}$$

using Fact 5.2.2 and a calculation. Thus we may take as our bound $U := 4^{-t}M^t$, so that by Eq. (5.4),

$$2^{t'+1} \;\ge\; .5 \cdot 4^t = 2^{2t-1},$$

which contradicts our assumption that $t' < 2(t-1)$. This proves Lemma 5.3.1. $\square$

Proof of Theorem 5.1.1. We will show that the existence of the reduction R for L implies that there exists a two-message, private-coin, *interactive proof system* between a polynomial-time-bounded Verifier and a computationally unbounded Prover to prove that a given string $x \in \{0,1\}^n$ lies in $\overline{L}$. The proof system will be executable using poly(n) bits of non-uniform advice on length-n inputs; Prover will be able to make Verifier accept with probability 1 if $x \notin L$, and with probability at most $1 - \Omega(1)$ if $x \in L$. It then follows from known results on interactive proof systems and non-uniform derandomization [GS86, Adl78] that $\overline{L} \in \mathsf{NP/poly}$ (see Theorem 5.4.11 and the proof of Theorem 5.4.15 for details), which gives our desired conclusion.

Using the existence of the reduction R and Lemma 5.3.1, we will prove the following claim:

Claim 5.3.2. *There exist multisets $A_1, \ldots, A_{q(n) \le \text{poly}(n)} \subseteq L_n$, each of size bounded by some $s(n) \le \text{poly}(n)$, such that, for all $x \in \{0,1\}^n \setminus \left(\bigcup_{i \in [q(n)]} A_i\right)$:*

1. *If $x \in \overline{L}_n$, then $\beta(x; A_i) = 1$ for all $i \in [q(n)]$;*
2. *If $x \in L_n$, there is an $i \in [q(n)]$ for which $\beta(x; A_i) \le 1 - 10^{-5}$.*

Assuming the truth of Claim 5.3.2 for the moment, we use it to prove Theorem 5.1.1. For inputs of length n to our interactive proof system, we let the non-uniform advice be a description of the sets $A_1, \ldots, A_{q(n)}$ given by Claim 5.3.2, along with the value $t(n)$. The proof system works as follows. On input $x \in \{0,1\}^n$, Verifier first checks if x is in one of the sets A_i. If so, Verifier knows that $x \in L$. Otherwise, Verifier and Prover execute the following procedure in parallel for $i = 1, 2, \ldots, q(n)$:

- Verifier privately flips an unbiased coin $b_i \sim \mathcal{U}_{\{0,1\}}$;
- Verifier privately samples strings $y^{i,1}, \ldots, y^{i,t(n)} \in \{0,1\}^n$ independently from $\mathcal{U}_{A_i}$;
- If $b_i = 0$ then Verifier sets

$$z \;=\; z(i) \;:=\; R(y^{i,1}, \ldots, y^{i,t(n)}) \;;$$

otherwise ($b_i = 1$), Verifier samples $\mathbf{j} = \mathbf{j}(i) \sim \mathcal{U}_{[t(n)]}$ and sets

$$z \ := \ R(y^{i,1}, \ldots, y^{i,\mathbf{j}-1}, x, y^{i,\mathbf{j}+1}, \ldots, y^{i,t(n)}) \ .$$

- Verifier sends z to Prover.
- Prover makes a guess $\widetilde{b}_i$ for the value of b_i.

Verifier accepts iff $\widetilde{b}_i = b_i$ for all i.

This protocol is clearly polynomial-time executable by Arthur given $t(n)$ and the description of $A_1, \ldots, A_{q(n)}$, and these sets are of polynomial size and polynomial in number. Now let us analyze the behavior of the protocol (assuming $x \notin \bigcup_i A_i$). First, suppose that $x \in \overline{L}_n$. In this case, we have

$$||\mathbf{R}_{A_i}[x] - \mathbf{R}_{A_i}||_{\text{stat}} \ = \ 1$$

for each i, by the first property of our sets A_i. Thus, Prover can guess b_i with perfect confidence for each i, and can cause Verifier to accept with probability 1.

Next, suppose that $x \in L_n$. Then by the second property of our sets, there exists an $i^* \in [q(n)]$ such that

$$||\mathbf{R}_{A_{i^*}}[x] - \mathbf{R}_{A_{i^*}}||_{\text{stat}} \ \leq \ 1 - 10^{-5} \ .$$

By the distinguishability characterization of statistical distance, and the independence of the trials $i = 1, 2, \ldots, q(n)$, this implies that the probability that Prover guesses b_{i^*} correctly is at most $1 - .5 \cdot 10^{-5}$. Thus Verifier rejects with probability $\Omega(1)$. So our interactive proof has the desired properties. As discussed earlier, this implies $\overline{L} \in \mathsf{NP/poly}$. $\square$

Proof of Claim 5.3.2. Fixing attention to a single value of n, let $(t, t') = (t(n), t'(n))$. Assume that t is large enough to apply Lemma 5.3.1. (Note that then t' satisfies the assumptions of that lemma as well.) Let $M := 100t$. We define a sequence of sets $S_1 \supseteq S_2 \supseteq \ldots \supseteq S_{q(n)+1} = \emptyset$, each contained in L_n, and a sequence of sets $A_1, A_2, \ldots, A_{q(n)}$, with all elements of A_i drawn from S_i.

Let $S_1 := L_n$. Inductively, having defined S_i, we define A_i, S_{i+1} as follows. If $|S_i| < M$, we let $A_i := S_i$ and $S_{i+1} := \emptyset$, and set $q(n) := i$, terminating the construction at this stage. Otherwise ($|S_i| \geq M$), we let A_i be a uniformly random size-$(M-1)$ subset of S_i. We let

$$S_{i+1} := \left\{ a \in S_i \setminus A_i : \beta(a, A_i) \ > \ 1 - 10^{-5} \right\} \ .$$

The procedure clearly terminates, since $|S_{i+1}| \leq |S_i| - (M-1)$ whenever $S_{i+1} \neq \emptyset$. Let us verify that these A_i satisfy conditions 1-2 of the Claim; we will then argue that $q(n) \leq \text{poly}(n)$ (with high probability over the randomness in the construction).

First, suppose $x \in \overline{L}_n \setminus \left(\bigcup_{i \in [q(n)]} A_i \right)$. Then with attention to Eq. (5.1), note that R always outputs an element of $\overline{L'}$ when x is one of the inputs to R. On the other hand, when all inputs to R are drawn from some $A_i \subseteq S_i \subseteq L_n$, R outputs an element of L'. Thus these two cases are perfectly distinguishable, and $\beta(x, A_i) = 1$ for each i, as needed.

Next suppose $x \in L_n \setminus \left(\bigcup_{i \in [q(n)]} A_i \right)$. Let $i \in [1, q(n)]$ be the unique index such that $x \in S_i \setminus S_{i+1}$. Then by the definitions, we have $\beta(x, A_i) = ||\mathbf{R}_{A_i}[x] - \mathbf{R}_{A_i}||_{\text{stat}} \leq 1 - 10^{-5}$.

Finally, we argue that $q(n) \leq \text{poly}(n)$ with high probability. Note that when we generate A_i as a uniform set of size $M-1$, we may equivalently generate A_i by first generating a uniform set $\widehat{A}_i \subseteq S_i$ of size M, then selecting a uniform element a^* of $\widehat{A}_i$ to discard to form A_i.

By Lemma 5.3.1, $\mathbb{E}_{a^*}[\beta(a^*, A_i)] \leq 1 - 10^{-4}$. Then with probability at least .9 over our randomness at this stage, a^* satisfies $\beta(a^*, A_i) \leq 1 - 10^{-5}$. But a^* is distributed as a uniform element of $S_i \setminus A_i$. Thus,

$$\mathbb{E}[|S_{i+1}|] \ \leq \ .1(|S_i| - |A_i|) \ .$$

Thus $q(n) = O(n)$ with high probability. This completes the proof of Claim 5.3.2. $\square$

5.4 Preliminaries II

Now we collect facts and definitions that will inform our work in the rest of the chapter as we prove more general results.

5.4.1 Information theory background

Recall from Section 5.2 that $H(\alpha)$ denotes the binary entropy function on $[0, 1]$. For a finitely-supported random variable Z, we let

$$H_{rv}(Z) \;:=\; \sum_{z \,\in\, \mathrm{supp}(Z)} -\Pr[Z = z] \log_2 \Pr[Z = z]$$

denote the Shannon entropy of Z. Then, for two possibly-dependent random variables Y, Z,

$$H_{rv}(Z|Y) \;:=\; \mathbb{E}_{y \sim Y}[H_{rv}(Z_{[Y=y]})] \;=\; H_{rv}((Y, Z)) - H_{rv}(Y)$$

denotes the *entropy of Z conditional on Y*. ($Z_{[Y=y]}$ denotes Z conditioned on the event $[Y = y]$.)

Fact 5.4.1. *For all X, Y, $H_{rv}((X, Y)) \le H_{rv}(X) + H_{rv}(Y)$ and $H_{rv}(X|Y) \le H_{rv}(X)$, with equality holding in each case iff X, Y are independent. Similarly, $H_{rv}(X|(Y, Z)) \le H_{rv}(X|Y)$.*

Definition 5.4.2 (Mutual information)**.** *The* mutual information *between random variables X, Y is defined as $I(X; Y) := H_{rv}(X) + H_{rv}(Y) - H_{rv}((X, Y))$.*

The next fact follows easily from the definitions.

Fact 5.4.3. *Mutual information obeys the following properties, for all random variables X, Y, Z:*

1. *$I(X; Y) = I(Y; X)$;*
2. *$I(X; (Y, Z)) = I(X; Y) + I((X, Y); Z) - I(Y; Z)$;*
3. *$I(X; (Y, Z)) \ge I(X; Y)$;*

4. $I(X;Z) = 0$ if X, Z are independent.

Lemma 5.4.4. *If $X^1, \ldots, X^t$ are independent, then*

$$I(Y;(X^1,\ldots,X^t)) \;\geq\; \sum_{j\in[t]} I(Y;X^j)\ .$$

Our proof of this standard claim follows steps in [Nay99a, p. 33].

Proof. We have

$$\begin{aligned} I(Y;(X^1,\ldots,X^t)) &= I(Y;X^t) + I((Y,X^t);(X^1,\ldots,X^{t-1})) - \underbrace{I(X^t;(X^1,\ldots,X^{t-1}))}_{=0,\text{ by Fact 5.4.3, item 4}} \\ &\geq\; I(Y;X^t) + I(Y;(X^1,\ldots,X^{t-1}))\ , \end{aligned}$$

where we used item 2 of Fact 5.4.3 in the first step, and items 1 and 3 in the second step. Iterating in this way gives the Lemma. □

The next definition is a useful, non-symmetric measure of difference between random variables.

Definition 5.4.5 (KL divergence)**.** *The (binary)* Kullback-Leibler divergence*, or* KL divergence *between random variables X, Y, is denoted $D_{\mathrm{KL}}(X||Y)$ and defined as*

$$D_{\mathrm{KL}}(X||Y) \;:=\; \sum_{x\in\mathrm{supp}(X)} \Pr[X=x]\cdot\log_2\left(\frac{\Pr[X=x]}{\Pr[Y=x]}\right)\ .$$

The convention is that for $p \neq 0$, we have $p\log_2(p/0) = +\infty$. So D_{KL} may be infinite. We have the following basic equivalence (see [CT06, Chapter 2]):

Fact 5.4.6. *Let X, Y be any random variables; let X' be distributed as X and independent of Y. The mutual information and Kullback-Leibler divergence satisfy*

$$I(X;Y) \;=\; D_{\mathrm{KL}}((X,Y)||(X',Y))\ .$$

A proof of the following important result can be found in [CT06] (see Lemma 11.6.1, p. 370).

Theorem 5.4.7 (Pinsker's inequality, stated for binary KL divergence). *For any random variables Z, Z',*

$$D(Z||Z') \;\geq\; \frac{2}{\ln 2} \cdot ||Z - Z'||^2_{\mathrm{stat}} \;.$$

When $||Z-Z'||_{\mathrm{stat}} \approx 1$, the following bound, known as Vajda's inequality (see [FHT03, RW09]), gives better information on the divergence:

Theorem 5.4.8 (Vajda's inequality, stated for binary KL divergence). *For any random variables Z, Z', let $\Delta := ||Z - Z'||_{\mathrm{stat}}$. Then,*

$$D(Z||Z') \;\geq\; \frac{1}{\ln 2}\left(\ln\left(\frac{1+\Delta}{1-\Delta}\right) - \frac{2\Delta}{1+\Delta}\right) \;\geq\; \frac{1}{\ln 2}\left(\ln\left(\frac{1}{1-\Delta}\right) - 1\right).$$

5.4.2 Basic complexity classes and promise problems

We assume familiarity with the basic complexity classes NP and coNP and the higher levels Σ_k^p, Π_k^p of the Polynomial Hierarchy PH. (For the needed background in complexity theory, see [AB09].) In this chapter we define $\mathsf{NP}, \mathsf{coNP}$, etc. as classes of *languages* (not promise problems).

We also assume familiarity with the general model of polynomial-size, non-uniform advice, and with the non-uniform classes $\mathsf{NP/poly}$ and $\mathsf{coNP/poly}$. It is considered unlikely that $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. In particular, this would imply a collapse of the Polynomial Hierarchy:

Theorem 5.4.9 ([Yap83]). *If* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, *then* $\mathsf{PH} = \Sigma_3^p = \Pi_3^p$.

We use $\mathsf{pr\text{-}NP}$, $\mathsf{pr\text{-}coNP}$, etc. to denote the analogous complexity classes for *promise problems.* Recall that, for a class C of promise problems, $\mathsf{coC} = \{(\Pi_Y, \Pi_N) : (\Pi_N, \Pi_Y) \in \mathsf{C}\}$. A *many-to-one* reduction B from the promise problem $\Pi = (\Pi_Y, \Pi_N)$ to $\Pi' = (\Pi'_Y, \Pi'_N)$ is a mapping satisfying $B(\Pi_Y) \subseteq \Pi'_Y, B(\Pi_N) \subseteq \Pi'_N$. (This definition applies as well to the special case where one or both of the promise problems are languages.) When we refer to NP-complete problems in this chapter, we mean problems complete under deterministic, polynomial-time many-to-one reducibility.

All of the results we prove in this chapter about limits of compression for *languages* L and language complexity classes readily extend to the setting of compression for promise problems (under the analogous definitions). However, for notational simplicity we will state our main results for languages, and will only use promise problems and promise classes where doing so helps to streamline our proofs and our result statements.

5.4.3 Arthur-Merlin protocols

We will make use of the model of (public-coin, two-round) *Arthur-Merlin protocols*. To be precise, these are protocols $\mathcal{P}$, defined by a deterministic polynomial-time predicate $A(x, r, w)$, which operate as follows. On an input x, visible to both a polynomial-time bounded verifier (Arthur) and to a computationally-unbounded prover (Merlin):

1. Arthur generates a uniformly random string r and sends it to Merlin;
2. Merlin sends a response string w to Arthur;
3. Arthur accepts if $A(x, r, w) = 1$, otherwise rejects.

We require that $|r|, |w|$ each be pre-specified lengths $\leq \text{poly}(n)$, where $n = |x|$, and that these lengths be computable in $\text{poly}(n)$ time given 1^n.

We will need to work with promise problems having Arthur-Merlin protocols. Say that such a protocol $\mathcal{P}$ *defines* a promise problem $\Pi = (\Pi_Y, \Pi_N)$ with *completeness* $c(n)$ *and soundness* $s(n)$ if

1. For all $x \in \Pi_Y$, some Merlin strategy causes Arthur to accept with probability $\geq c(n)$;
2. For all $x \in \Pi_N$, all Merlin strategies cause Arthur to accept with probability $\leq s(n)$.

Let $\mathsf{pr\text{-}AM}_{c(n),s(n)}$ denote the class of promise problems definable by an Arthur-Merlin protocol with completeness $c(n)$ and soundness $s(n)$; let $\mathsf{pr\text{-}AM} := \mathsf{pr\text{-}AM}_{1,1/3}$. Then, $\mathsf{pr\text{-}coAM} = \{(\Pi_Y, \Pi_N) : (\Pi_N, \Pi_Y) \in \mathsf{pr\text{-}AM}\}$.

Theorem 5.4.10 ([FGM$^+$89]). *For any parameters $s(n), c(n) \in (0,1]$ that are polynomial-time computable*[23] *and satisfy $\frac{1}{\mathrm{poly}(n)} < s(n) < c(n) - \frac{1}{\mathrm{poly}(n)}$, we have $\mathsf{pr\text{-}AM}_{c(n),s(n)} = \mathsf{pr\text{-}AM}$. If we drop the requirement $s(n) > \frac{1}{\mathrm{poly}(n)}$, but keep the gap requirement, we still have $\mathsf{pr\text{-}AM}_{c(n),s(n)} \subseteq \mathsf{pr\text{-}AM}$.*

The next, well-known result follows from the non-uniform derandomization technique of Adleman [Adl78]:

Theorem 5.4.11. $\mathsf{pr\text{-}AM} \subseteq \mathsf{pr\text{-}NP/poly}$*. Similarly,* $\mathsf{pr\text{-}coAM} \subseteq \mathsf{pr\text{-}coNP/poly}$.

5.4.4 Statistical zero-knowledge and the SD problem

Next we will define the *statistical zero-knowledge* class SZK. Actually, we will only work with its promise-problem analogue $\mathsf{pr\text{-}SZK}$.[24] Informally, these are the promise problems (Π_Y, Π_N) for which a (private-coin) interactive proof of membership in Π_Y can be given, in which the verifier *learns (almost) nothing—except* to become convinced that the input y indeed lies in Π_Y! The "learns nothing" requirement is cashed out by requiring that the verifier be able to *simulate* interactions with the intended prover strategy on any input y, such that if $y \in \Pi_Y$, the resulting distribution is negligibly close in statistical distance to the true distribution generated by their interaction.

Making this definition formal is somewhat delicate. (For details, and for more information on these and related classes, see [SV03].) Fortunately, there is a simple (but non-trivial) characterization of $\mathsf{pr\text{-}SZK}$. First, given a Boolean circuit $C = C(r)$ with k output gates, and an ordering on these gates, let $\mathcal{D}_C$ denote the output distribution of C on a uniformly random input r. (This is a random variable over $\{0,1\}^k$.) We use the following problem:

Definition 5.4.12. *For parameters $0 \le d \le D \le 1$, define the promise problem $\mathrm{SD}^{\ge D}_{\le d} = (\Pi_Y, \Pi_N)$ as follows:*

$$\Pi_Y \ := \ \{\langle C, C'\rangle : ||\mathcal{D}_C - \mathcal{D}_{C'}||_{\mathrm{stat}} \ge D\} \ ,$$

[23](say, as rational values represented by their numerator and denominator)

[24]Often the promise class is denoted SZK.

$$\Pi_N \ := \ \{\langle C, C'\rangle : ||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}} \leq d\} \ .$$

Define $\mathrm{SD}^{\leq d}_{>D}$ *analogously, switching the "yes" and "no" cases. In this definition, both* $d = d(n)$ *and* $D = D(n)$ *may be parameters depending on the input length* $n = |\langle C, C'\rangle|$.

It is shown in [SV03] that the standard, complicated definition of $\mathsf{pr\text{-}SZK}$ is equivalent to the following simpler one, which we take as our definition:

Definition 5.4.13. *Let* $\mathsf{pr\text{-}SZK}$ *be defined as the class of promise problems for which there is a many-to-one,*[25] *deterministic polynomial-time reduction from* Π *to* $\mathrm{SD}^{\geq 2/3}_{\leq 1/3}$.

The constants $2/3, 1/3$ in the above definition are not arbitrary; it is unknown whether we get the same class if we replace them by $.51, .49$. However, we have the following result:

Theorem 5.4.14 (Follows from [SV03]; described as Theorem 1 in [GV11])**.** *Suppose* $0 \leq d = d(n) < D = D(n) \leq 1$ *are polynomial-time computable, and satisfy* $D^2 > d + \frac{1}{\mathrm{poly}(n)}$. *Then,* $\mathrm{SD}^{\geq D}_{\leq d} \in \mathsf{pr\text{-}SZK}$.

When we merely have $D - d \geq \frac{1}{\mathrm{poly}(n)}$, the following weaker, standard result holds:

Theorem 5.4.15. *Suppose* $0 \leq d = d(n) < D = D(n) \leq 1$ *are polynomial-time computable and satisfy* $D > d + \frac{1}{\mathrm{poly}(n)}$. *Then,* $\mathrm{SD}^{\geq D}_{\leq d} \in \mathsf{pr\text{-}AM}$.

Proof sketch. We describe a *private-coin* two-message protocol, in which the verifier has a source of random bits not viewable by the prover; any such protocol can be efficiently converted into a public-coin one [GS86].

Let $m = m(n) \leq \mathrm{poly}(n)$ be a large value. On input $\langle C, C'\rangle$, Verifier chooses $b_1, \ldots, b_m$ uniformly at random and, for $i \in [m]$, samples

$$z^i \sim \mathcal{D}_C \qquad \text{if } b_i = 0, \qquad z^i \sim \mathcal{D}_{C'} \quad \text{if } b_i = 1,$$

independently for each i. Prover is asked to try to guess the values $b_1, \ldots, b_m$.

[25]Recall the definition in Section 5.4.2.

If m is chosen appropriately large then, using the distinguishability interpretation of statistical distance (see Section 5.2.1),

1. If $||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}} > D$ then Prover can, with high probability, guess at least a $\frac{1}{2}\left(1 + \frac{D-d}{2}\right)$ fraction of the bits b_i correctly;

2. If $||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}} < d$ then Prover cannot, except with low probability, guess this fraction of the b_is correctly.

Thus, Verifier can use this threshold as an acceptance criterion, so that the protocol has the desired completeness-soundness gap. After converting to a public-coin protocol, we find that $\text{SD}^{\geq D}_{\leq d} \in \mathsf{pr\text{-}AM}_{2/3,1/3} = \mathsf{pr\text{-}AM}$ (using Theorem 5.4.10). □

We will also use the following important results about $\mathsf{pr\text{-}SZK}$:

Theorem 5.4.16 ([Oka00]). $\mathsf{pr\text{-}SZK}$ *is closed under complement.*

Theorem 5.4.17. $\mathsf{pr\text{-}SZK} \subseteq \mathsf{pr\text{-}AM} \cap \mathsf{pr\text{-}coAM} \subseteq \mathsf{pr\text{-}NP/poly} \cap \mathsf{pr\text{-}coNP/poly}$.

The containment in $\mathsf{pr\text{-}coAM}$ is due to Fortnow [For87]; containment in $\mathsf{pr\text{-}AM}$ was first shown by Aiello and Håstad [AH91].[26] The second containment in Theorem 5.4.17 uses Theorem 5.4.11.

Finally, one of our results (Theorem 5.7.3) will make use of the class $\mathsf{pr\text{-}PZK}$ of problems having (honest-verifier) *perfect* zero-knowledge proofs. This is a subclass of $\mathsf{pr\text{-}SZK}$. We will not define $\mathsf{pr\text{-}PZK}$ (see, e.g., [SV03]); unfortunately it has no known simple characterization analogous to Definition 5.4.13 for $\mathsf{pr\text{-}SZK}$. We will, however, use the following result:

Theorem 5.4.18 ([SV03], Proposition 5.7). $\text{SD}^{\geq 1}_{\leq .5} \in \mathsf{pr\text{-}PZK}$.

Next we combine tools described in Sections 5.4.3 and 5.4.4, reformulating them slightly.

[26]These works treat language classes, but the proofs extend without change to the promise-problem setting. Also, these works analyze a so-called "honest-verifier" model of statistical zero-knowledge proofs; these were shown to have the same expressive power as "cheating-verifier" statistical zero-knowledge proofs in [GSV98].

Theorem 5.4.19. *Let $0 \leq d = d(n) < D = D(n) \leq 1$ be (not necessarily computable) parameters.*

1. *If $D > d + \frac{1}{\text{poly}(n)}$, then $\text{SD}^{\geq D}_{\leq d} \in \textsf{pr-NP/poly}$.*
2. *If we have the stronger gap $D^2 > d + \frac{1}{\text{poly}(n)}$, then $\text{SD}^{\geq D}_{\leq d}$ is many-to-one reducible to $\text{SD}^{\geq 2/3}_{\leq 1/3} \in \textsf{pr-SZK}$, in non-uniform polynomial time. Also, $\text{SD}^{\geq D}_{\leq d} \in \textsf{pr-coNP/poly}$.*

Proof sketch. For item 1, we essentially combine Theorem 5.4.15 with Theorem 5.4.11. The only extra ingredient needed is to encode sufficiently accurate approximations of $d(n), D(n)$ into the non-uniform advice for length n, and to use these in defining the private-coin protocol as in the proof of Theorem 5.4.15. We then convert this non-uniform protocol into an $\textsf{NP/poly}$ one by the same techniques from [GS86] (which shows how to convert private-coin to public-coin protocols), Theorem 5.4.10 (to get perfect completeness), and Theorem 5.4.11 (to derandomize).

Similarly, for item 2, we essentially combine Theorems 5.4.14 and 5.4.17, except that at each step we need to incorporate approximations of $d(n), D(n)$ as (additional) non-uniform advice. □

5.4.5 f-compression reductions

Here we define a class of compression reductions for the problems $f \circ L$ introduced in Section 5.1.3, in which one is given $(x^1, \ldots, x^m)$ and must compute $f(L(x^1), \ldots, L(x^m))$. Our main focus will be the case where f is the OR or AND function of its input bits. The problem $f \circ L$ will be formally defined as a parametrized problem in Section 5.5.1, but it will be useful to have a specialized definition for this problem as well; here we won't explicitly rely on the parametrized-problem framework.

Our next definition is modeled on definitions in [BDFH09, FS11], with some differences. Notably, we will consider reductions where a quantitative compression guarantee is only made when all the input strings x^j are of some equal length n, and the number of input strings x^j is equal to some value $t_1(n)$ determined by n. The error bound will also be a function of n. This specialization is mostly to reduce clutter in our work, and will not lead to loss of generality: we will be *ruling out* the existence of compression reductions (under complexity-theoretic assumptions, and for all $t_1(n)$ that are sufficiently large compared to other parameters), so ruling out even compression algorithms that work only in narrow input-regimes will lead to stronger results.

Definition 5.4.20 (Probabilistic f-compression reductions)**.** *Let L, L' be two languages, and let $f : \{0,1\}^* \to \{0,1\}$ be a Boolean function. Let $t_1(n), t_2(n) : \mathbb{N}^+ \to \mathbb{N}^+$ and $\xi(n) : \mathbb{N}^+ \to [0,1]$ be given.*

A probabilistic f-compression reduction for L*, with parameters $(t_1(n), t_2(n), \xi(n))$ and target language L', is a randomized mapping $R(x^1, \ldots, x^m)$ outputting a string z, such that for all $(x^1, \ldots, x^{t_1(n)}) \in \{0,1\}^{t_1(n) \times n}$,*

1. $\Pr_R[L'(z) = f\left(L(x^1), \ldots, L(x^{t_1(n)})\right)] \geq 1 - \xi(n)$*;*

2. $|z| \leq t_2(n)$*.*

If some reduction R as above is computable in probabilistic polynomial time, we say that L is PPT-f-compressible *with parameters $(t_1(n), t_2(n), \xi(n))$. (This does not require that $(t_1(n), t_2(n), \xi(n))$ themselves be computable.)*

5.5 Parametrized problems and parametrized compression

A central aim of our work is to better understand the limitations of efficient compressive reductions for a variety of parametrized problems. For this we need to formally define parametrized problems and an appropriate model of probabilistic compression for these problems. However, some readers may be satisfied to understand our work on the limits of efficient AND- and OR-compression (as defined in Section 5.4.5) for SAT and other NP-complete languages. To prove these results, including Theorem 5.1.3 in the Introduction, we will not need the definitions of this section, and readers may choose to skip ahead to Section 5.6. (We will find it convenient to prove Theorem 5.1.2 using the definitions below; however, this result can also be derived directly from our Theorem 5.7.1, item 2 with little trouble.)

5.5.1 Parametrized problems

We will use the following definition:

Definition 5.5.1 ([DF99])**.** *A* parametrized problem *is a subset of binary strings of the form $\langle x, 1^k \rangle$, for $x \in \{0,1\}^*$ and $k > 0$ (under some natural binary encoding of such tuples).*

Thus, our convention is that a parametrized problem is just a particular type of decision problem, i.e., a language.[27] However, we will use P to denote a generic parametrized problem, as opposed to an "ordinary" language, denoted L. Sometimes, as in the Introduction, we speak of "parametrized versions" of an ordinary decision problem L. There is no single, canonical way to go from a decision problem to a parametrized problem; often, however, a parametrized problem can be formed from a decision problem L in a natural way. For example, we formally define VAR-SAT, OR(SAT), and AND(SAT) from the Introduction as follows:

Definition 5.5.2. *Fix some natural encoding of tuples of bit-strings, and some encoding of Boolean formulas as bit-strings. Define*

1. VAR-SAT $:= \{\langle \psi, 1^k \rangle \mid \psi$ *is satisfiable and contains* $\leq k$ *distinct variables*$\}$;

2. OR(SAT) $:= \{\langle \psi_1, \ldots, \psi_t, 1^k \rangle \mid$ *at least one* ψ_j *is satisfiable, and each* ψ_j *is of bit-length* $\leq k\}$;

3. AND(SAT) $:= \{\langle \psi_1, \ldots, \psi_t, 1^k \rangle \mid$ *every* ψ_j *is satisfiable, and each* ψ_j *is of bit-length* $\leq k\}$.

We also generalize items 2 and 3 above:

Definition 5.5.3. *Let $L \subseteq \{0,1\}^*$, and $f : \{0,1\}^* \to \{0,1\}$. Define*

[27]In this definition we are following [FS11]. In [BDFH09] and many other works, parametrized problems are defined as a subset of $\{0,1\}^* \times \mathbb{N}^+$ (the parameter is still presented as part of the input); they refer to the corresponding subset of strings of form $\langle x, 1^k \rangle$ as the "unparametrized version" or "classical version" of the problem.

1. $\mathrm{OR}(L) := \{\langle (x^1, \ldots, x^t), 1^k \rangle \mid \bigvee_{j=1}^{t} L(x^i) = 1$ *and* $|x^j| \leq k$ *for each* $j\}$;

2. $\mathrm{AND}(L) := \{\langle (x^1, \ldots, x^t), 1^k \rangle \mid \bigwedge_{j=1}^{t} L(x^i) = 1$ *and* $|x^j| \leq k$ *for each* $j\}$;

3. $f \circ L := \{\langle (x^1, \ldots, x^t), 1^k \rangle \mid f(L(x^1), \ldots, L(x^t)) = 1$ *and* $|x^j| \leq k$ *for each* $j\}$.

5.5.2 OR-expressive and AND-expressive parametrized problems

Our compression lower bounds will apply to two classes of parametrized problems. As we will explain, these classes are closely related to classes identified earlier in [HN10, BDFH09, BJK11a, BTY11]; the classes we introduce will help to apply our techniques uniformly to these various earlier classes.

Definition 5.5.4 (OR- and AND-expressive problems)**.** *A parametrized problem P is* OR-expressive*, with parameter $S(n) \leq \mathrm{poly}(n)$, if there exists an* NP*-complete language L and a deterministic polynomial-time reduction B. Whenever B receives an input of form $\langle (x^1, \ldots, x^t), 1^n \rangle$, for any $t, n \in \mathbb{N}^+$, B outputs a tuple*

$$(\langle y^1, 1^{k_1} \rangle, \ldots, \langle y^s, 1^{k_s} \rangle) \ .$$

We have the following properties:

1. $\langle (x^1, \ldots, x^t), 1^n \rangle \ \in \ \mathrm{OR}(L) \quad \Longleftrightarrow \quad \exists\, i \in [s] : \langle y^i, 1^{k_i} \rangle \ \in P$;

2. $s \leq S(n)$ *(in particular, the bound is independent of t)*;

3. *For each $i \in [s]$, $|y^i| \leq (t+n)^{O(1)}$ and $k_i \leq n^{O(1)}$.*

Define AND-expressive *problems identically, except we replace condition 1 above by*

1'. $\langle (x^1, \ldots, x^t), 1^n \rangle \ \in \ \mathrm{AND}(L) \quad \Longleftrightarrow \quad \forall\, i \in [s] : \langle y^i, 1^{k_i} \rangle \ \in P$.

The results of [BDFH09] imply that a variety of natural parametrized problems are OR- or AND-expressive:

Theorem 5.5.5 (Follows from [BDFH09]). *1.* OR(SAT) *is* OR-*expressive with* $S(n) = 1$*. Also, each of the following parametrized problems are* OR-*expressive with* $S(n) \leq \mathrm{poly}(n)$*:*

- k-Path*,* k-Cycle*,* k-Exact Cycle *and* k-Short Cheap Tour*,*
- k-Graph Minor Order Test *and* k-Bounded Treewidth Subgraph Test*,*
- k-Planar Graph Subgraph Test *and* k-Planar Graph Induced Subgraph Test*,*
- (k, σ)-Short Nondeterministic Turing Machine Computation*,*
- w-Independent Set*,* w-Clique *and* w-Dominating Set*,*

defined in [BDFH09].

2. AND(SAT) *is* AND-*expressive with* $S(n) = 1$*. Also, each of the following parametrized problems are* AND-*expressive with* $S(n) \leq \mathrm{poly}(n)$*:*

- k-Cutwidth*,* k-Modified Cutwidth*, and* k-Search Number*,*
- k-Pathwidth*,* k-Treewidth*, and* k-Branchwidth*,*
- k-Gate Matrix Layout *and* k-Front Size*,*
- w-*3*-Coloring *and* w-*3*-Domatic Number*,*

also defined in [BDFH09].

In [BDFH09], the authors define a notion of *compositionality* for parametrized problems. If a parametrized problem P is compositional and NP-complete, then it is OR-expressive, with respect to the NP-complete language $L = P$. Also, if P is NP-complete and $\overline{P}$ is compositional, then P is AND-expressive. These facts follow almost immediately from the definitions. Theorem 5.5.5 then follows from the compositionality results proved in [BDFH09]. In a number of the problems above we can actually take $S(n) = 1$.

Bodlaender, Jansen, and Kratsch [BJK11a] introduced a notion of *cross-compositionality* of parametrized problems, generalizing compositionality. They showed that the evidence against efficient compression against compositional problems given by [BDFH09, FS11] can be extended to cross-compositional problems. Cross-compositional problems are also OR-expressive, as follows from the definitions [BJK11a, Section 3].[28] As shown in [BJK11a], this class includes interesting parametrized versions of the Clique, Chromatic Number, and Feedback Vertex Set problems.

AND-expressiveness results are fewer in number, although this is partly due to the fact that, after the results of [FS11] appeared, OR-expressiveness results were preferentially sought. Another example of an AND-expressive problem (not known to be OR-expressive) is presented in [BJK11c].

We also have the following result, derived from the earlier work of [HN10]:

Theorem 5.5.6 (Follows from [HN10, FS11])**.** *Each of the problems* Clique*,* Dominating Set*,*[29] Integer Programming*, described in [HN10] and modeled as parametrized problems in [FS11] (with slightly distinctive, but natural, parametrizations), are* OR*-expressive, with $S(n) = 1$.*

A class of reductions between parametrized problems, called *W-reductions*, is used in these works (see [FS11, Definition 2.10]); OR(SAT) is shown to W-reduce to each of the problems listed in Theorem 5.5.6. This immediately implies that these problems are OR-expressive with $S(n) = 1$. Also, if an OR-expressive parametrized problem P W-reduces to a second problem Q, then Q is also OR-expressive. This technique was used in [BTY11] to derive additional hardness-of-compression results for problems not easily captured by the compositionality framework; our new results apply to these problems as well.

We remark that the polynomial bounds involved in the reductions of Theorems 5.5.5 and 5.5.6 are fairly modest.

[28]Strictly speaking, according to their definition, cross-compositional problems are OR-expressive under the minor restriction on the reduction in Definition 5.5.4 that the input $\langle(x^1, \ldots, x^t), 1^n\rangle$ satisfy $t \leq 2^{n^a}$, for some $a > 0$. This is of no importance to us, since we will always work with the case $t \leq \text{poly}(n)$; we could have required this in Definition 5.5.4, and could prove the same variety of hardness results.

[29](these are different parametrized problems than w-Clique and w-Dominating Set in Theorem 5.5.5 above)

5.5.3 Parametrized compression

We define compression reductions for parametrized problems as follows, following [FS11] (but with some added flexibility in our definitions):

Definition 5.5.7 (Probabilistic parametrized compression reductions)**.** *Let P be a parametrized problem and L' be a language, and say we are given two functions*

$$c(m,k,w) : (\mathbb{N}^+)^3 \ \to \ \mathbb{N}^+, \quad \xi(m,k,w) : (\mathbb{N}^+)^3 \ \to \ [0,1] \ .$$

*Say that a randomized mapping $R : \{0,1\}^* \to \{0,1\}^*$ is a (c,ξ)-*parametrized compression reduction *for P, with target language L', if for all inputs of form $\langle y, 1^k, 1^w\rangle$, $R(\langle y, 1^k, 1^w\rangle)$ outputs a string z such that:*

1. $\Pr_R[L'(z) = P(\langle y, 1^k\rangle)] \geq 1 - \xi(|y|, k, w)$;

2. $|z| \leq c(|y|, k, w)$.

We call c the compression bound *and ξ the* error bound *of the reduction; we call w the* confidence parameter.

For a parametrized problem P, if some reduction R as above is computable in probabilistic polynomial time, we say that P is PPT-compressible *with parameters (c,ξ).*

We will not be exploring the full range of possible parameter values in the above definition, but we believe it provides a reasonable framework for future work. (Only a few interesting examples of randomized parametrized compression reductions seem to be known; see [HN10, KW12].) The idea of a confidence parameter w, that one can use to increase the reliability of the compression at the expense of a potentially larger output size, is natural for probabilistic compression and will be useful in our work. (The same basic notion was used earlier in [FS11].)

Next, we define a notion of "strong" compressibility as in the Introduction, preserving flexibility in the error bound:

Definition 5.5.8. *Say that P is* strongly PPT-compressible *with error bound* $\xi(m, k, w)$, *if P is PPT-compressible (to some target language L') with error bound ξ and some compression bound c satisfying $c(m, k, 1) \leq k^{O(1)}$, with the polynomial bound independent of m.*

Using the majority-vote technique of [FS11, Proposition 5.1], we have the following easy result:

Lemma 5.5.9. *Let $a > 0$. Suppose that P is strongly PPT-compressible with error bound satisfying $\xi(m, k, 1) \leq .5 - k^{-O(1)}$. Then, P is also PPT-compressible with compression bound $c'(m, k, w) \leq k^{O(1)} \cdot w$ and error bound $\xi'(m, k, w) \leq 2^{-w}$.*

5.5.4 Connecting parametrized compression and f-compression

The next lemma shows that to give evidence against efficient compression for "expressive" parametrized problems, it suffices to give evidence against efficient AND- and OR-compression for NP-complete languages. This lemma is modeled on [BDFH09, Lemma 2], but with some slight complications due to the probabilistic setting. For simplicity we only treat strong compression in the result below; our techniques also extend to give evidence against more modest compression amounts for expressive problems. (For more modest compression amounts, the obtainable results are weaker when the parameter $S(n)$ in the definition of expressiveness is fast-growing.)

Lemma 5.5.10. *Let L be an NP-complete language.*

1. *Suppose that the parametrized problem P is* OR*-expressive with respect to L, with parameter $S(n) \leq \text{poly}(n)$. If P is strongly PPT-compressible with error bound $\xi(m, k, 1) \leq .5 - k^{-O(1)}$, then for any polynomially-bounded function $T(n) : \mathbb{N}^+ \to \mathbb{N}^+$, L is PPT-*OR*-compressible with parameters*

$$t_1(n) = T(n), \quad t_2(n) \leq S(n) \cdot n^{O(1)}, \quad \xi'(n) \leq 2^{-n} .$$

2. *Suppose P is* AND*-expressive with respect to L. If P is strongly PPT-compressible*

*with error bound $\xi(m, k, 1) \leq .5 - k^{-O(1)}$, then L is PPT-*AND*-compressible with parameters $(t_1(n), t_2(n), \xi'(n))$ as in item 1.*

Proof of Lemma 5.5.10. We will prove item 1 above; item 2 is proved similarly. Let R be the PPT compression reduction R for P given by Lemma 5.5.9. Let L'_0 be the target language of R. Let B be the reduction for P and L as in Definition 5.5.4.

We define an OR-compression reduction R' for L, with target language $L' := \text{OR}(L'_0)$, as follows. In defining R', we let $t_1(n) := T(n)$. On inputs $x^1, \ldots, x^{T(n)} \in \{0,1\}^{T(n)\times n}$, the reduction first applies B to $\langle (x^1, \ldots, x^{T(n)}), 1^n \rangle$, yielding a tuple $(\langle y^1, 1^{k_1}\rangle, \ldots, \langle y^s, 1^{k_s}\rangle)$. Next, for each $i \in [s]$, R' applies R to the string $\langle y^i, 1^{k_i}, 1^{2n}\rangle$ (here we are selecting the confidence parameter $w := 2n$ for R), yielding an output z^i. Then R' outputs $\langle (z^1, \ldots, z^s), 1^M\rangle$, where $M := \max_i |z^i|$.

R' is clearly polynomial-time computable. Now let us analyze its compression and reliability properties. First, each y^i is of bit-length $|y^i| \leq (T(n) + n)^{O(1)}$, and $k_i \leq n^{O(1)}$, by item 3 of Definition 5.5.4. Then by the compression guarantee for R, each z^i is of bit-length $\leq n^{O(1)} \cdot w = n^{O(1)}$. Thus for the output-size bound of R' we may take $t_2(n) \leq S(n) \cdot n^{O(1)}$, as needed.

Now we bound the error of R'. Using the correctness property of B (Definition 5.5.4, item 1), the equivalence

$$\langle (z^1, \ldots, z^s), 1^M\rangle \in \text{OR}(L'_0) \iff \bigvee_{j=1}^{T(n)} [x^j \in L]$$

holds as long as each application of R, namely $R(\langle y^i, 1^{k_i}\rangle)$ for $i \in [s]$, is successful. By a union bound, this occurs with probability $\geq 1 - S(n) \cdot 2^{-2n}$, which is larger than $1 - 2^{-n}$ for sufficiently large n. (For smaller n, R' may solve its input problem directly by brute force.) Thus for the error bound $\xi'(n)$ for R', we may take $\xi'(n) \leq 2^{-n}$. $\square$

5.6 Technical lemmas

In this section we present our main technical lemmas. Our final goal in this section will be the "Disguising-Distribution Lemma," our key technical tool for our main results.

5.6.1 Distributional stability

Here we define the notion of "distributional stability" described in Section 5.1.4.

Definition 5.6.1. *Let U be some finite universe, and let $T, n \geq 1$ be integers. Given a possibly-randomized mapping $F(x^1, \ldots, x^T) : \{0,1\}^{T \times n} \to U$, and a collection $\mathcal{D}_1, \ldots, \mathcal{D}_T$ of mutually independent distributions over $\{0,1\}^n$, for $j \in [T]$ let*

$$\gamma_j \ := \ \mathop{\mathbb{E}}_{y \sim \mathcal{D}_j} \left[||F(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, y, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t) - F(\mathcal{D}_1, \ldots, \mathcal{D}_t)||_{\text{stat}} \right] \ .$$

For $\delta \in [0,1]$, say that F is δ-distributionally stable (or δ-DS) with respect to *$\mathcal{D}_1, \ldots, \mathcal{D}_T$ if*

$$\frac{1}{T} \sum_{j=1}^{T} \gamma_j \ \leq \ \delta \ .$$

Lemma 5.6.2. *Let $R(x^1, \ldots, x^t) : \{0,1\}^{t \times n} \to \{0,1\}^{\leq t'}$ be any possibly-randomized mapping, for any $n, t, t' \in \mathbb{N}^+$. R is δ-distributionally stable with respect to any independent input distributions $\mathcal{D}_1, \ldots, \mathcal{D}_t$, where we may take either of the following two bounds:*

1. $\delta := \sqrt{\frac{\ln 2}{2} \cdot \frac{t'+1}{t}}$;
2. $\delta := 1 - 2^{-\frac{t'}{t} - 3}$.

Our proof of Lemma 5.6.2, item 1 essentially follows suggestions by Ashwin Nayak and Salil Vadhan; item 2 is a small modification using Vajda's inequality. When $t'/t = 1 - \Omega(1)$, the bound given in item 1 above is within constant factors of the bound from our original distributional stability lemma, Lemma 5.10.4. On the other hand, when $t'/t = 1 - \alpha \approx 1$, the bound in Lemma 5.6.2, item 1 is better (i.e., smaller) by a $\Theta\left(\log \frac{1}{\alpha}\right)$ factor. We don't know how to prove a version of item 2 above with the methods of Lemma 5.10.4; this alternative bound is important for our work. In an earlier draft we used a more complicated workaround to prove the results obtainable from item 2.

Proof of Lemma 5.6.2. Define independent random variables $X^j \sim \mathcal{D}_j$ over $\{0,1\}^n$, for $j \in [t]$. Let $\mathbf{R} := R(X^1, \ldots, X^t)$.

The entropy of $\mathbf{R}$ is at most $\log_2\left(\left|\{0,1\}^{\leq t'}\right|\right) < t' + 1$. Thus, the mutual information $I((X^1, \ldots, X^t); \mathbf{R})$ is less than $t' + 1$. By the independence of the X^js, Lemma 5.4.4 gives

$$\sum_{j \in [t]} I(X^j; \mathbf{R}) \ < \ t' + 1 \ . \tag{5.6}$$

By Fact 5.4.6,

$$I(X^j; \mathbf{R}) \ = D_{\mathrm{KL}}\left((X^j, \mathbf{R}) \,||\, (Y^j, \mathbf{R})\right) \ , \tag{5.7}$$

where $Y^j \sim \mathcal{D}_j$ is independent of $\mathbf{R}$. By Theorem 5.4.7,

$$\begin{aligned} D_{\mathrm{KL}}\left((X^j, \mathbf{R}) \,||\, (Y^j, \mathbf{R})\right) \ &\geq \ \frac{2}{\ln 2} \cdot ||(X^j, \mathbf{R}) - (Y^j, \mathbf{R})||_{\mathrm{stat}}^2 \\ &= \ \frac{2}{\ln 2} \cdot \mathbb{E}_{x^j \sim \mathcal{D}_j}\left[\left|\left|R\left(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, x^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t\right) - R\left(\mathcal{D}_1, \ldots, \mathcal{D}_t\right)\right|\right|_{\mathrm{stat}}\right]^2 \end{aligned}$$

where the equality follows from the distinguishability interpretation of statistical distance. Using this, we find

$$\begin{aligned} &\left(\frac{1}{t}\sum_{j \in [t]} \mathbb{E}_{x^j \sim \mathcal{D}_j}\left[\left|\left|R\left(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, x^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t\right) - R\left(\mathcal{D}_1, \ldots, \mathcal{D}_t\right)\right|\right|_{\mathrm{stat}}\right]\right)^2 \\ &\leq \ \frac{1}{t}\sum_{j \in [t]} \mathbb{E}_{x^j \sim \mathcal{D}_j}\left[\left|\left|R\left(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, x^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t\right) - R\left(\mathcal{D}_1, \ldots, \mathcal{D}_t\right)\right|\right|_{\mathrm{stat}}\right]^2 \\ &\text{(by Jensen's inequality)} \\ &< \ \frac{\ln 2}{2} \cdot \frac{t' + 1}{t} \ . \end{aligned}$$

Thus, R is $\sqrt{\frac{\ln 2}{2} \cdot \frac{t'+1}{t}}$-distributionally stable with respect to $\mathcal{D}^1, \ldots, \mathcal{D}^t$. This proves item 1 of the Lemma.

For item 2, we apply the alternative bound, Vajda's inequality (Theorem 5.4.8), to each $j \in [t]$, to find

$$D_{\mathrm{KL}}\left((X^j, \mathbf{R}) \,||\, (Y^j, \mathbf{R})\right) \ \geq \ \frac{1}{\ln 2}\left(\ln\left(\frac{1}{1 - ||(X^j, \mathbf{R}) - (Y^j, \mathbf{R})||_{\mathrm{stat}}}\right) - 1\right)$$

$$= \frac{1}{\ln 2}\left(\ln\left(\frac{1}{\varepsilon_j}\right) - 1\right) \ ,$$

where we define

$$\varepsilon_j := 1 - \mathbb{E}_{x^j \sim \mathcal{D}_j}\left[\left|\left|R\left(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, x^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t\right) - R\left(\mathcal{D}_1, \ldots, \mathcal{D}_t\right)\right|\right|_{\text{stat}}\right]$$

and note that $\varepsilon_j > 0$. Averaging over $j \in [t]$ and applying Eqs. (5.6) and (5.7),

$$\frac{t'+1}{t} \ \geq \ \frac{1}{t}\sum_{j \in [t]} \frac{1}{\ln 2}\left(\ln\left(\frac{1}{\varepsilon_j}\right) - 1\right) \ ,$$

i.e.,

$$\frac{1}{t}\sum_{j \in [t]} \ln\left(\frac{1}{\varepsilon_j}\right) \ \leq \ \frac{(\ln 2)(t'+1)}{t} + 1 \ .$$

The function $f(x) = \ln(1/x)$ has second derivative $x^{-2} > 0$ for $x > 0$, and so Jensen's inequality gives

$$\ln\left(\frac{1}{\frac{1}{t}\sum_{j \in [t]} \varepsilon_j}\right) \ \leq \ \frac{(\ln 2)(t'+1)}{t} + 1 \ .$$

This implies

$$\frac{1}{t}\sum_{j \in [t]} \varepsilon_j \ \geq \ \left(e^{\frac{(\ln 2)(t'+1)}{t} + 1}\right)^{-1} \ \geq \ 2^{-\frac{t'}{t} - 3} \ ,$$

which proves item 2. □

5.6.2 Sparsified distributional stability

Here we prove a technical lemma showing that if a mapping F is distributionally stable with respect to i.i.d. inputs, then F also obeys a slightly different stability property, in which we replace an input distribution $\mathcal{D}$ with a "sparsified" version of $\mathcal{D}$.

Lemma 5.6.3. *Let U be a finite set, and let $F(x^1, \ldots, x^T) : \{0,1\}^{T \times n} \to U$ be given. Suppose F is δ-distributionally stable with respect to input distribution $\mathcal{D}^{\otimes T}$, for every distribution $\mathcal{D}$ over $\{0,1\}^n$.*

Fix some distribution $\mathcal{D}$ over $\{0,1\}^n$, and let $x^1, \ldots, x^d$ be independently sampled from $\mathcal{D}$. Let $k^* \sim \mathcal{U}_{[d]}$.

Let $\widehat{\mathcal{D}}$ denote the distribution defined by sampling uniformly from the multiset $\{x^k\}_{k \neq k^*}$. (This distribution is itself a random variable, determined by $x^1, \ldots, x^d$ and by k^*.) Define

$$\beta_j \;:=\; \mathop{\mathbb{E}}_{k^*, x^1, \ldots, x^d} \left[\; \left\| F\left(\widehat{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widehat{\mathcal{D}}^{\otimes(T-j)}\right) \;-\; F\left(\widehat{\mathcal{D}}^{\otimes T}\right) \right\|_{\text{stat}} \; \right] \; ,$$

where all the $\widehat{\mathcal{D}}$s are to be mutually independent (for fixed values of $x^1, \ldots, x^d$ and k^*). Then,

$$\frac{1}{T} \sum_{j=1}^{t} \beta_j \;\le\; \delta + 2T/d \; .$$

Proof. Let $\widetilde{\mathcal{D}}$ denote the distribution, determined by $x^1, \ldots, x^d$, that samples uniformly from the multiset $\{x^k\}_{k \in [d]}$. By an easy calculation, for any values of $x^1, \ldots, x^d$ and k^* we can bound

$$\left\| \widetilde{\mathcal{D}} - \widehat{\mathcal{D}} \right\|_{\text{stat}} \;\le\; 1/d \; .$$

It follows that

$$\left\| F\left(\widetilde{\mathcal{D}}^{\otimes T}\right) - F\left(\widehat{\mathcal{D}}^{\otimes T}\right) \right\|_{\text{stat}} \;\le\; \left\| \widetilde{\mathcal{D}}^{\otimes T} - \widehat{\mathcal{D}}^{\otimes T} \right\|_{\text{stat}} \;\le\; T/d \; ,$$

where in the last step we used Fact 5.2.5 and the fact that for any assignment to $x^1, \ldots, x^d$ and to k^*, the T copies of $\widetilde{\mathcal{D}}$ used are mutually independent, as are the copies of $\widehat{\mathcal{D}}$.

By identical reasoning, for any assignment to $x^1, \ldots, x^d$ and to k^*, and for any index $j \in [T]$ we have

$$\left\| F\left(\widetilde{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widetilde{\mathcal{D}}^{\otimes(T-j)}\right) - F\left(\widehat{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widehat{\mathcal{D}}^{\otimes(T-j)}\right) \right\|_{\text{stat}} \;\le\; (T-1)/d \; .$$

Using the triangle inequality for $|| \cdot ||_{\text{stat}}$, for any values $x^1, \ldots, x^d, k^*$ and any index

$j \in [T]$ we always have

$$\begin{aligned}
&\left\|F\left(\widehat{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widehat{\mathcal{D}}^{\otimes(T-j)}\right) \;-\; F\left(\widehat{\mathcal{D}}^{\otimes T}\right)\right\|_{\text{stat}} \\
&\le\; \left\|F\left(\widehat{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widehat{\mathcal{D}}^{\otimes(T-j)}\right) - F\left(\widetilde{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widetilde{\mathcal{D}}^{\otimes(T-j)}\right)\right\|_{\text{stat}} \\
&\quad+\; \left\|F\left(\widetilde{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widetilde{\mathcal{D}}^{\otimes(T-j)}\right) \;-\; F\left(\widetilde{\mathcal{D}}^{\otimes T}\right)\right\|_{\text{stat}} \\
&\quad+\; \left\|F\left(\widetilde{\mathcal{D}}^{\otimes T}\right) - F\left(\widehat{\mathcal{D}}^{\otimes T}\right)\right\|_{\text{stat}} \\
&\le\; \left\|F\left(\widetilde{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widetilde{\mathcal{D}}^{\otimes(T-j)}\right) \;-\; F\left(\widetilde{\mathcal{D}}^{\otimes T}\right)\right\|_{\text{stat}} + 2T/d \;. \qquad (5.8)
\end{aligned}$$

Now suppose we fix any values $x^1, \ldots, x^d$, leaving k^* undetermined. The value k^* is uniform on $[d]$, so that x^{k^*} is distributed exactly according to $\widetilde{\mathcal{D}}$. Under our conditioning, let

$$\gamma_j = \gamma_j\left(\{x^k\}_{k\in[d]}\right) \;:=\; \mathop{\mathbb{E}}_{k^*}\left[\; \left\|F\left(\widetilde{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widetilde{\mathcal{D}}^{\otimes(T-j)}\right) \;-\; F\left(\widetilde{\mathcal{D}}^{\otimes T}\right)\right\|_{\text{stat}} \;\right] \;.$$

By our original assumption, F is δ-DS with respect to input distribution $\widetilde{\mathcal{D}}^{\otimes T}$. Thus, for any $x^1, \ldots, x^d$ we have

$$\frac{1}{t}\sum_{j=1}^{t} \gamma_j \;\le\; \delta \;. \qquad (5.9)$$

Now γ_j is itself a random variable, determined by $x^1, \ldots, x^d$, and from Eq. (5.8) we have

$$\beta_j \;\le\; \mathbb{E}[\gamma_j] + 2T/d \;.$$

Using linearity of expectation, we find that

$$\frac{1}{t}\sum_{j=1}^{t} \beta_j \;\le\; \delta + 2T/d \;.$$

□

5.6.3 Building disguising distributions

In the next lemmas we show how the distributional stability of a mapping F can be used to obtain a "disguising distribution" for F. In Lemma 5.6.6 we will apply this to give disguising distributions for any sufficiently compressive mapping R.

Recall that $\mathcal{U}_K$ denotes the uniform distribution over a multiset K.

Lemma 5.6.4. *Suppose $F(x^1, \ldots, x^T) : \{0,1\}^{T \times n} \to U$ obeys the assumption of Lemma 5.6.3: namely, F is δ-distributionally stable with respect to input distribution $\mathcal{D}^{\otimes T}$, for every distribution $\mathcal{D}$ over $\{0,1\}^n$.*

Let $S \subseteq \{0,1\}^n$, and fix some value $d > 0$. There exists a distribution $\mathcal{K}$ over size-d multisets $K \subseteq S$, such that for every $y \in S$, the following holds:

$$\mathop{\mathbb{E}}_{K \sim \mathcal{K}, j^* \sim \mathcal{U}_{[T]}} \left[\left|\left| F\left(\mathcal{U}_K^{\otimes(j^*-1)}, y, \mathcal{U}_K^{\otimes(T-j^*)}\right) \;-\; F\left(\mathcal{U}_K^{\otimes T}\right) \right|\right|_{\text{stat}} \right] \;\leq\; \delta + 2T/(d+1) \;.$$

(Here the copies of $\mathcal{U}_K$ are to be mutually independent for fixed K, although the set $K \sim \mathcal{K}$ used is the same for each copy.)

Proof. Consider the following two-player, simultaneous-move, zero-sum game:

- **Player 1:** chooses a size-d multiset $K \subseteq S$.
- **Player 2:** chooses a string $y \in S$.
- **Payoff:** Player 2 receives a payoff equal to

$$\mathop{\mathbb{E}}_{j^* \sim \mathcal{U}_{[T]}} \left[\left|\left| F\left(\mathcal{U}_K^{\otimes(j^*-1)}, y, \mathcal{U}_K^{\otimes(T-j^*)}\right) \;-\; F\left(\mathcal{U}_K^{\otimes T}\right) \right|\right|_{\text{stat}} \right] \;.$$

  (Note that this payoff is a determinate value, given (K, y).)

Consider any randomized strategy by Player 2, specified by a distribution $y \sim Y$ over S. In response, let $\mathcal{K}_Y$ be the randomized Player-1 strategy that chooses a size-d multiset K of elements sampled independently from Y.

To bound the expected payoff under the strategy-pair $(\mathcal{K}_Y, Y)$, note that we can equivalently generate $(K, y) \sim (\mathcal{K}_Y, Y)$ as follows. First, sample $x^1, \ldots, x^{d+1}$ independently from Y. Sample $k^* \sim \mathcal{U}_{[d+1]}$, set $y := x^{k^*}$, and let

$$K := \{x^1, \ldots, x^{k^*-1}, x^{k^*+1}, \ldots, x^{d+1}\} .$$

It is easily verified that $(K, y) \sim (\mathcal{K}_Y, Y)$ as desired.

Then Lemma 5.6.3, applied to our initial distributional-stability assumption on F, informs us that

$$\mathop{\mathbb{E}}_{j^* \sim \mathcal{U}_{[t]}, K, y} \left[\ \left\| F\left(\mathcal{U}_K^{\otimes(j^*-1)}, y, \mathcal{U}_K^{\otimes(T-j^*)}\right) \ - \ F\left(\mathcal{U}_K^{\otimes T}\right) \right\|_{\mathrm{stat}} \ \right] \ \leq \ \delta + 2T/(d+1) \ .$$

Thus Player 2's expected payoff against $\mathcal{K}_Y$ is at most $\delta + 2T/(d+1)$.

As Y was arbitrary, the minimax theorem tells us that there exists a distribution $\mathcal{K}$ over Player-1 moves that forces Player 2's expected payoff under *every* strategy to be at most $\delta + 2T/(d+1)$. The result follows. $\square$

Lemma 5.6.5. *Let U be a finite set, and let $F(x^1, \ldots, x^T) : \{0,1\}^{T\times n} \to U$ be given. Suppose F is δ-distributionally stable with respect to input distribution $\mathcal{D}^{\otimes T}$, for every distribution $\mathcal{D}$ over $\{0,1\}^n$.*

Let $S \subseteq \{0,1\}^n$, and fix $d > 0$. Given any $\varepsilon > 0$, let $s := \lceil (.5 \ln 2) n/\varepsilon^2 \rceil$. Then there exists a collection $K_1, \ldots, K_s$ of size-d multisets contained in S, such that for every $y \in S$ the following holds:

$$\mathop{\mathbb{E}}_{a \sim \mathcal{U}_{[s]}, j^* \sim \mathcal{U}_{[t]}} \left[\ \left\| F\left(\mathcal{U}_{K_a}^{\otimes(j^*-1)}, y, \mathcal{U}_{K_a}^{\otimes(T-j^*)}\right) \ - \ F\left(\mathcal{U}_{K_a}^{\otimes T}\right) \right\|_{\mathrm{stat}} \ \right] \ \leq \ \delta + 2T/(d+1) + \varepsilon \ .$$

Proof. This is an immediate application (to the game in Lemma 5.6.4) of a general result due to Lipton and Young [LY94, Theorem 2], showing that all two-player, zero-sum games have sparsely-supported, nearly-optimal player strategies. (Essentially the same result was proved independently by Althöfer [Alt94], and a more general result for many-player, non-zero-sum games was proved later in [LMM03].) The support size required in the Lipton-Young-Althöfer result depends logarithmically on the number of pure strategies available to the player we are opposing; in our case, Player 2 has a choice of $|S| \leq 2^n$ strings y, so we get $s = O(n/\varepsilon^2)$. In their proof technique applied to our setting, the $K_1, \ldots, K_s$ are obtained by sampling independently from the distribution $\mathcal{K}$ given by Lemma 5.6.4, giving a successful outcome with nonzero probability. □

Lemma 5.6.6 (Disguising-Distribution Lemma). *Let $R(x^1, \ldots, x^t) : \{0,1\}^{t\times n} \to \{0,1\}^{\leq t'}$ be any possibly-randomized mapping, for $t, t' \in \mathbb{N}^+$. Let $S \subseteq \{0,1\}^n$, and fix $d > 0$. Given any $\varepsilon > 0$, let $s := \lceil (.5 \ln 2) n/\varepsilon^2 \rceil$. Let*

$$\widehat{\delta} := \min \left\{ \sqrt{\frac{\ln 2}{2} \cdot \frac{t'+1}{t}}, \quad 1 - 2^{-\frac{t'}{t} - 3} \right\} .$$

Then there exists a collection $K_1, \ldots, K_s$ of size-d multisets contained in S, such that for every $y \in S$, we have

$$\mathop{\mathbb{E}}_{a \sim \mathcal{U}_{[s]}, j^* \sim \mathcal{U}_{[t]}} \left[\left\| R\left(\mathcal{U}_{K_a}^{\otimes(j^*-1)},\ y,\ \mathcal{U}_{K_a}^{\otimes(t-j^*)}\right) \ - \ R\left(\mathcal{U}_{K_a}^{\otimes t}\right) \right\|_{\mathrm{stat}} \right]$$
$$\leq \ \widehat{\delta} + 2t/(d+1) + \varepsilon \ .$$

Proof. This follows immediately from the combination of Lemmas 5.6.2 and 5.6.5, applied to $F := R$ (and with $T := t$). □

5.7 Limits to efficient (classical) compression

In this section, we show that a sufficiently high-quality PPT-OR-compression reduction for any language L implies that $L \in \mathsf{NP/poly}$. We also show that above a higher threshold of quality, such a compression reduction implies that L has non-uniform, statistical zero-knowledge proofs, which in particular implies $L \in \mathsf{coNP/poly}$ as well. We will then apply these results to give evidence against efficient probabilistic compression for AND(SAT) and OR(SAT), as described in the Introduction, and for other parametrized problems with either of the two "expressiveness" properties described in Section 5.5.2. We will also present our result on f-compression reductions for more general combining functions f, and our result extending the work of Dell and Van Melkebeek [DvM10] on problems with polynomial kernelizations.

5.7.1 Complexity upper bounds from OR-compression schemes

Theorem 5.7.1. *Let L be any language. Suppose $t_1(n), t_2(n) : \mathbb{N}^+ \to \mathbb{N}^+$ are (not necessarily computable) functions. Suppose that there exists a PPT-*OR*-compression reduction $R(x^1, \ldots, x^t) : \{0,1\}^{t_1(n)\times n} \to \{0,1\}^{\leq t_2(n)}$ for L with parameters $t_1(n), t_2(n)$, error bound $\xi(n) < .5$, and some target language L'. Let*

$$\widehat{\delta} := \min\left\{\sqrt{\frac{\ln 2}{2}\cdot\frac{t_2(n)+1}{t_1(n)}},\quad 1 - 2^{-\frac{t_2(n)}{t_1(n)}-3}\right\} .$$

1. *If for some constant $c > 0$ we have*

$$1 - 2\xi(n) - \widehat{\delta} \;\geq\; \frac{1}{n^c} \;, \tag{5.10}$$

   then $L \in \mathsf{NP/poly}$.

2. *If for some $c > 0$ we have the (stronger) bound*

$$(1 - 2\xi(n))^2 - \widehat{\delta} \;\geq\; \frac{1}{n^c} \;, \tag{5.11}$$

   then there is a many-to-one reduction from L to a promise problem in $\mathsf{pr\text{-}SZK}$. The reduction is computable in non-uniform polynomial time; in particular, this implies $L \in \mathsf{NP/poly} \cap \mathsf{coNP/poly}$.

We remark that, using the technique of [FS11, Proposition 5.1], one can reduce the error bound $\xi(n)$ of an OR-compression scheme, at the cost of increasing the output-length bound $t_2(n)$. (The idea is to perform multiple, independent applications of R to the fixed input tuple $(x^1, \ldots, x^{t_1(n)})$ and to concatenate the results in the output, using a majority-vote rule to define a new target language.) With this amplification, we can in some cases apply Theorem 5.7.1 where its assumptions do not hold for the original scheme—or, we may obtain the stronger conclusion in item 2 of Theorem 5.7.1 in cases where only item 1 would apply directly.

Proof of Theorem 5.7.1. We will use the same basic reduction to prove items 1 and 2.

First, with non-uniformity it is easy to handle length-n inputs whenever $L_n = \{0,1\}^n$, so let us assume from this point on that $\overline{L}_n$ is nonempty.

Using R, we define a deterministic, non-uniform polynomial-time reduction $\mathcal{R}$ that, on input $y \in \{0,1\}^n$, builds a description of two circuits C, C'. The aim is that $||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}}$ should be large if $y \in L$, and small if $y \notin L$. $\mathcal{R}$ works as follows:

- **Non-uniform advice for length n:** a description of the value $t_1(n)$, and the multisets $K_1, \ldots, K_s \subseteq \overline{L}_n$ given by Lemma 5.6.5 with

  $$(t, t') \ := \ (t_1(n), t_2(n)), \quad S := \overline{L}_n, \quad d := \lceil 8t_1(n) \cdot n^c \rceil \,, \quad \varepsilon := \frac{1}{4n^c} \ .$$

  (Here $c > 0$ is as in Eq. (5.10) or Eq. (5.11), according to which item of the Theorem we are proving.) Note that d and the value s given by Lemma 5.6.5 are both $\leq \text{poly}(n)$ under these settings, so our advice is of polynomial length.

- **On input $y \in \{0,1\}^n$:** let $\mathcal{R}$ output descriptions $\langle C, C' \rangle$ of the following two randomized circuits:

  - **Circuit C:** samples $a \sim \mathcal{U}_{[s]}$, then samples

    $$\overline{x} \ = \ (x^1, \ldots, x^{t_1(n)}) \ \sim \ \mathcal{U}_{K_a}^{\otimes t_1(n)} \ ,$$

    and outputs $z := R(\overline{x})$.

  - **Circuit C':** samples values

    $$a \sim \mathcal{U}_{[s]}, \quad j^* \sim \mathcal{U}_{[t_1(n)]} \ ;$$

    then, samples

    $$\overline{x} \ \sim \ \left(\mathcal{U}_{K_a}^{\otimes (j^*-1)}, \ y, \ \mathcal{U}_{K_a}^{\otimes (t_1(n) - j^*)} \right) \ ,$$

    and outputs $z := R(\overline{x})$.

Claim 5.7.2. *The following holds:*

1. *If $y \in L$, then*
$$||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}} \;\geq\; D(n) \;:=\; 1 - 2\xi(n) \;;$$

2. *If $y \notin L$, then*
$$||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}} \;\leq\; d(n) \;:=\; \widehat{\delta} + \frac{1}{2n^c} \;. \tag{5.12}$$

We defer the proof of Claim 5.7.2, and use it to prove the two items of Theorem 5.7.1.

For item 1 of Theorem 5.7.1, if Eq. (5.10) holds (for sufficiently large n), then $D(n) - d(n) \geq \frac{1}{n^c}$.

Now $D(n), d(n)$ were parametrized in terms of $n = |y|$, but the gap $D(n) - d(n)$ is also at least inverse-polynomial in the length $N \leq \text{poly}(n)$ of the output description $\langle C, C' \rangle$. Thus our reduction $\mathcal{R}$ reduces any instance y of the decision problem for L, to an equivalent instance $\mathcal{R}(y) = \langle C, C' \rangle$ of the promise problem $\text{SD}^{\geq D'(N)}_{\leq d'(N)}$, with different parameters $D'(N), d'(N)$ still satisfying the gap condition $D' - d' \geq \frac{1}{\text{poly}(N)}$.

By item 1 of Theorem 5.4.19, $\text{SD}^{\geq D'}_{\leq d'} \in \textsf{pr-NP/poly}$. Let $(A, \{a_N\}_{N>0})$ be an nondeterministic, non-uniform polynomial-time algorithm and advice family solving $\text{SD}^{\geq D'}_{\leq d'}$. Then by applying $(A, \{a_N\})$ to $\mathcal{R}(y)$, we obtain a nondeterministic, non-uniform polynomial-time algorithm for solving L. This shows $L \in \textsf{NP/poly}$, proving item 1 of the Theorem.

Next, for item 2 of Theorem 5.7.1, if Eq. (5.11) holds for sufficiently large n, then $D(n)^2 - d(n) \geq \frac{1}{n^c}$. Arguing as in the previous case, but this time applying item 2 of Theorem 5.4.19, we exhibit a nonuniform polynomial-time reduction from L to $\text{SD}^{\geq D'}_{\leq d'}$, where this time $D'(N)^2 - d'(N) \geq \frac{1}{\text{poly}(N)}$. This problem can in turn be reduced to $\text{SD}^{\geq 2/3}_{\leq 1/3} \in \textsf{pr-SZK}$ in non-uniform polynomial time, by the second assertion of Theorem 5.4.14. This also yields $L \in \textsf{NP/poly} \cap \textsf{coNP/poly}$, and completes the proof of Theorem 5.7.1. □

Proof of Claim 5.7.2. **(1.)** First, suppose $y \in L$. We will use the distinguishing interpretation of statistical distance (see Section 5.2.1) to argue that $||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}}$ is large. Suppose an unbiased coin $b \sim \mathcal{U}_{\{0,1\}}$ is flipped, unseen by us, and we receive a sample $z \sim \mathcal{D}_C$ if $b = 0$, or $z \sim \mathcal{D}_{C'}$ if $b = 1$. Consider the distinguisher that outputs the guess $\tilde{b} := 0$ if $z \in \overline{L'}$, or $\tilde{b} :=$ if $z \in L'$.

We lower-bound the success probability $\Pr[\tilde{b} = b]$ as follows. Say we condition on $[b = 0]$, so that $z \sim \mathcal{D}_C$. The distributions $\mathcal{U}_{K_a}$ are supported on $\overline{L}_n$, so in the execution of C we get $\overline{x} \in \left(\overline{L}_n\right)^{t_1(n)}$. Then it follows from the OR-compression property of R for L that $\Pr[z \in \overline{L'}] \geq 1 - \xi(n)$. On the other hand, suppose we condition on $[b = 1]$, so that $z \sim \mathcal{D}_{C'}$. In an execution of C' the input tuple $\overline{x}$ contains $y \in L_n$; thus, by the OR-compression property of R, we have $\Pr[z \in L'] \geq 1 - \xi(n)$. So regardless of the value of b, our distinguisher succeeds with probability $\geq 1 - \xi(n)$. Thus, $1 - \xi(n) \leq \frac{1}{2}(1 + ||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}})$. This proves item 1.

(2.) Now suppose $y \notin L$; we must upper-bound $||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}}$. Consider the distinguishing experiment between C and C' as in item 1. If we regard the random variables a and j^* (the latter used only by C') to be part of the joint probability space of both algorithms (noting that a is identically distributed in the two circuits), then revealing the values a, j^* along with z to the distinguisher cannot decrease the distinguisher's maximum achievable success probability. Now conditioned on revealed values a, j^*, the maximum achievable success probability in the modified distinguishing experiment is

$$\frac{1}{2}\left(1 + \left|\left|R\left(\mathcal{U}_{K_a}^{\otimes t_1(n)}\right) - R\left(\mathcal{U}_{K_a}^{\otimes(j^*-1)},\ y,\ \mathcal{U}_{K_a}^{\otimes(t_1(n)-j^*)}\right)\right|\right|_{\text{stat}}\right) ,$$

from which we conclude that

$$||\mathcal{D}_C - \mathcal{D}_{C'}||_{\text{stat}} \leq \mathop{\mathbb{E}}_{a\sim\mathcal{U}_{[s]}, j^*\sim\mathcal{U}_{[t_1(n)]}}\left[\ \left|\left|R\left(\mathcal{U}_{K_a}^{\otimes t_1(n)}\right) - R\left(\mathcal{U}_{K_a}^{\otimes(j^*-1)},\ y,\ \mathcal{U}_{K_a}^{\otimes(t_1(n)-j^*)}\right)\right|\right|_{\text{stat}}\ \right] . \tag{5.13}$$

By our choice of $K_1, \ldots, K_s$ and Lemma 5.6.6, the right-hand side of Eq. (5.13) is at most

$$\widehat{\delta} + 2t_1(n)/(d+1) + \varepsilon \ < \ \widehat{\delta} + 2 \cdot \frac{1}{4n^c} , \tag{5.14}$$

by our settings to d, ε. This proves Eq. (5.12) and completes the proof of Claim 5.7.2.

□

The next result gives a useful consequence of Theorem 5.7.1 for the case where the compression bound $t_2(n)$ is on the order of $t_1(n) \cdot \log_2(t_1(n))$, and also points out a strengthening of the result's conclusion in the case of error-free compression.

Theorem 5.7.3. *Let L be any language. Suppose $t_1(n), t_2(n) : \mathbb{N}^+ \to \mathbb{N}^+$ satisfy $t_2(n) \leq C \cdot t_1(n) \log t_1(n)$ and $t_1(n) \leq n^{C'}$, for some $C, C' > 0$. Suppose that R is a PPT-*OR*-compression reduction $R(x^1, \ldots, x^{t_1(n)}) : \{0,1\}^{t_1(n) \times n} \to \{0,1\}^{\leq t_2(n)}$ for L with parameters $t_1(n), t_2(n)$, error bound $\xi(n) < .5$, and some target language L'.*

1. *If $\xi(n) < n^{-C \cdot C'}/32$, then there is a non-uniform polynomial-time many-to-one reduction from L to a promise problem in* pr-SZK.

2. *Suppose further that R is error-free (i.e., $\xi(n) = 0$). Then, there is a non-uniform polynomial-time many-to-one reduction from L to a promise problem in* pr-PZK.

Proof. **(1.)** We bound the quantity $\widehat{\delta}$ from Theorem 5.7.1:

$$\begin{aligned}
\widehat{\delta} &\leq 1 - 2^{-\frac{t_2(n)}{t_1(n)} - 3} \\
&\leq 1 - 2^{-C \log_2(t_1(n))}/8 \\
&\leq 1 - t_1(n)^{-C}/8 \\
&\leq 1 - n^{-C \cdot C'}/8 \ .
\end{aligned}$$

If $\xi(n) < n^{-C \cdot C'}/32$, then the left-hand quantity in Eq. (5.11) is $\geq \frac{1}{\text{poly}(n)}$, and the desired conclusion then follows from Theorem 5.7.1, item 2.

(2.) Looking into the proof of Theorem 5.7.1, item 2, we see that it gives a non-uniform polynomial-time many-to-one reduction from L to $\text{SD}^{\geq D'(N)}_{\leq d'(N)}$, where in the current case, using Claim 5.7.2, we have

$$D'(N) = 1, \quad d'(N) \leq 1 - \frac{1}{\text{poly}(N)} \ .$$

This problem can in turn be uniformly many-to-one reduced to $\mathrm{SD}^{\geq 1}_{\leq .5}$ by mapping a circuit-distribution pair $\langle C, C' \rangle$ to $\langle C^{\otimes T}, (C')^{\otimes T} \rangle$, where $C^{\otimes T}$ is the circuit that outputs T samples drawn independently from C, and where $T \leq \mathrm{poly}(n)$ is chosen suitably large. Finally, $\mathrm{SD}^{\geq 1}_{\leq .5} \in \mathsf{pr\text{-}PZK}$ by Theorem 5.4.18. $\square$

5.7.2 Application to AND- and OR-compression of NP-complete languages

Throughout this section, for parameters $t_1(n), t_2(n)$, we will use the shorthand

$$\widehat{\delta} \;:=\; \min\left\{ \sqrt{\frac{\ln 2}{2} \cdot \frac{t_2(n)+1}{t_1(n)}}, \quad 1 - 2^{-\frac{t_2(n)}{t_1(n)} - 3} \right\}$$

Here is our first main result giving evidence against efficient AND-compression for NP-complete languages:

Theorem 5.7.4. *Suppose that for some NP-complete language L, any target language L', and an error bound $\xi(n) < .5$, L has a PPT-AND-compression reduction R with target language L', with parameters $t_1(n), t_2(n) : \mathbb{N}^+ \to \mathbb{N}^+$ and error bound $\xi(n) < .5$.*

1. *If*
$$1 - 2\xi(n) - \widehat{\delta} \;\geq\; \frac{1}{\mathrm{poly}(n)} \;, \tag{5.15}$$
then $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ and $\mathsf{PH} = \Sigma_3^p = \Pi_3^p$.

2. *If we have the bound*
$$(1 - 2\xi(n))^2 - \widehat{\delta} \;\geq\; \frac{1}{\mathrm{poly}(n)} \;, \tag{5.16}$$
then L (and every other language in NP) is many-to-one reducible in non-uniform polynomial time to a problem in $\mathsf{pr\text{-}SZK}$, and $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

3. *The conclusion of item 2 holds if $t_2(n) \leq C \cdot t_1(n) \log(t_1(n)))$ and if $\xi(n)$ is a sufficiently small inverse-polynomial function of n (determined by t_1 and the constant C).*

Item 3 above establishes the assertion of Theorem 5.1.3 from the Introduction for the case of AND-compression.

Proof of Theorem 5.7.4. **(1.)** The reduction R is also a PPT-OR-compression for $\overline{L}$, with target language $\overline{L'}$, and with the same parameters.

If Eq. (5.15) holds in case 1, we apply item 1 of Theorem 5.7.1 to $\overline{L}$, concluding that $\overline{L} \in \mathsf{NP/poly}$, i.e., $L \in \mathsf{coNP/poly}$. The consequence for PH is from Theorem 5.4.9.

(2.) Similarly, if Eq. (5.16) holds in case 2, we apply item 2 of of Theorem 5.7.1 to $\overline{L}$, giving a non-uniform many-to-one reduction from $\overline{L}$ to a problem $\Pi = (\Pi_Y, \Pi_N) \in \mathsf{pr\text{-}SZK}$. This is also a reduction from L to (Π_N, Π_Y), which by Theorem 5.4.14 also lies in $\mathsf{pr\text{-}SZK}$. The extension to other languages in NP follows from the NP-completeness of L.

(3.) In this case we apply Theorem 5.7.3, item 1 to $\overline{L}$. □

The next theorem gives evidence for the infeasability of efficient OR-compression for NP-complete languages.

Theorem 5.7.5. *1. Suppose that for some NP-complete language L, any target language L', and an error bound $\xi(n) < .5$, L has a PPT-OR-compression reduction R with target language L', with parameters $t_1(n), t_2(n) : \mathbb{N}^+ \to \mathbb{N}^+$ and error bound $\xi(n) < .5$. If*

$$(1 - 2\xi(n))^2 - \widehat{\delta} \geq \frac{1}{\mathrm{poly}(n)} , \tag{5.17}$$

then L (and every other language in NP) is reducible in non-uniform polynomial time to a problem in $\mathsf{pr\text{-}SZK}$, and $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

2. The conclusion of item 1 holds if $t_2(n) \leq C \cdot t_1(n) \log(t_1(n)))$ and if $\xi(n)$ is a sufficiently small inverse-polynomial function of n (determined by t_1 and C).

Item 2 completes the proof of Theorem 5.1.3 from the Introduction.

Proof of Theorem 5.7.5. **(1.)** This time, if Eq. (5.17) holds, we just apply item 2 of Theorem 5.7.1 to L itself.

(2.) In this case we apply item 1 of Theorem 5.7.3 to L. □

NOTE: As we noted in a correction to the chapter Intro, Subsection 5.7.3 contains a bug and Thm. 5.7.6 is wrong. Consult the ECCC revision of the paper for a corrected result.

5.7.3 f-compression of NP-complete languages for general f

As discussed in Section 5.1.3, our techniques, combined with ideas of [FS11, Section 7], imply some limitations to efficient strong f-compression of SAT or other NP-complete languages, for any "reasonable" combining function f.

Theorem 5.7.6. *Suppose $f : \{0,1\}^* \to \{0,1\}$ depends on all coordinates, and suppose that R is a strong f-compression reduction for an* NP*-complete language L with a target language $L' \in \mathsf{NP}$, computable in deterministic*[30] *polynomial time. Then* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

Proof sketch. Suppose the strong f-compression reduction maps inputs $(x^1, \ldots, x^t) \in \{0,1\}^{t\times n}$ to an output z of length at most n^c (independent of t); we have

$$z \in L' \iff f(L(x^1), \ldots, L(x^t)) = 1 \ .$$

Fix an input length n, and let $N := n^{2c+1}$. If f is non-monotone on inputs of size N, say with respect to the first coordinate, then we can non-uniformly fix some $N-1$ strings $x^2, \ldots, x^N \in \{0,1\}^n$ such that for $x \in \{0,1\}^n$,

$$R(x, x^2, \ldots, x^N) \in L' \iff L(x) = 0 \ .$$

This reduces the membership question for $\overline{L}_n$ to a membership question for $L' \in \mathsf{NP}$.

Otherwise, f is monotone for length-N inputs. Then by results of Nisan [Nis91], we can "embed" an AND or OR of size $\sqrt{N}$ into f_N. That is, there exists some subset $S \subseteq [N]$ of size at least $\sqrt{N} = n^{c+.5}$, and an assignment to all inputs to f outside of S, such that the restricted function is an AND or OR of its remaining

[30](this assumption is for simplicity)

coordinates. Thus, there exists a collection $(x^\ell)_{\ell \notin S}$ of length-n inputs to R, such that for any setting to the remaining (length-n) inputs with indices in S,

$$R(x^1, x^2, \ldots, x^N) \in L' \quad \Longleftrightarrow \quad \mathrm{OP}_{\ell \in S}\ (L(x^\ell)) = 1\ ,$$

where here OP is either AND or OR. Thus for this input length n, we have (non-uniformly) either an AND-compression or OR-compression reduction from L to target L', compressing by a polynomial amount. In either case, we can apply the techniques of Section 5.7.2 to get a non-uniform proof system for membership in $\overline{L}_n$. Combining our work for each input length, we find that $\overline{L} \in \mathsf{NP/poly}$. As L is NP-complete, the Theorem is proved. □

(End caution)

5.7.4 Limits to strong compression for parametrized problems

Next, we use Theorem 5.7.1 to give evidence against strong compressibility for "expressive" parametrized problems. The result we give below is a simple-to-state, representative example; the quantitative settings studied here are not the only interesting ones our techniques can handle.

Theorem 5.7.7. *Say that P is* OR*-expressive or* AND*-expressive, e.g., one of the problems listed in Theorems 5.5.5 and 5.5.6. Suppose additionally that P is strongly PPT-compressible*[31] *with error bound $\xi(m, k, w)$ satisfying $\xi(m, k, 1) \leq .5 - k^{-O(1)}$ (independent of m), i.e., with success probability $\geq .5 + k^{-O(1)}$. Then, every language in* NP *is many-to-one reducible in non-uniform polynomial time to a problem in* $\mathsf{pr\text{-}SZK}$ *(and $\mathsf{NP} \subseteq \mathsf{coNP/poly}$).*

Theorem 5.1.2 from the Introduction follows, by considering the special cases $P = \mathrm{OR(SAT)}$ and $P = \mathrm{AND(SAT)}$.

Proof of Theorem 5.7.7. Suppose first that P is OR-expressive, with respect to the NP-complete language L and with some parameter $S(n) \leq \mathrm{poly}(n)$. We apply item

[31](as in Definition 5.5.8)

1 of Lemma 5.5.10 to L and the assumed strong compression reduction for P. Using some function $T(n) \leq \text{poly}(n)$ to be determined, and with $w(n) := 1$, we obtain a PPT-OR-compression for L with parameters

$$t_1(n) \ = \ T(n), \quad t_2(n) \ \leq \ S(n) \cdot n^{O(1)}, \quad \xi'(n) \ \leq \ 2^{-n} \ .$$

(Here, the bound on t_2 is independent of the choice of $T(n)$.) We evaluate

$$(1 - 2\xi'(n))^2 - \sqrt{\frac{\ln 2}{2} \cdot \frac{t_2(n) + 1}{t_1(n)}} \quad \geq \quad (1 - 4 \cdot 2^{-n}) - \sqrt{\frac{\ln 2}{2} \cdot \frac{S(n) \cdot n^{O(1)} + 1}{T(n)}} \ ,$$

for some $a > 0$ (using $S(n) \leq \text{poly}(n)$). The expression above can be made greater than .5 for large n by choosing a sufficiently fast-growing $T(n) \leq \text{poly}(n)$. Under such a setting, Eq (5.17) holds for $(t_1(n), t_2(n), \xi'(n))$. We can then apply the first assertion of Theorem 5.7.5, item 1 to our PPT-OR-compression for L, which yields the desired conclusion.

The case where P is AND-expressive is handled analogously; in this case we apply Lemma 5.5.10, item 2 and the first assertion of Theorem 5.7.4, item 2. □

We can also apply Theorem 5.7.3 to show that, if any NP-complete language L is PPT-OR-compressible by an error-free reduction with $t_2(n) = O(t_1(n) \log(t_1(n)))$, then NP has non-uniform *perfect* zero-knowledge proofs. From a deterministic AND-compression reduction for L of this type, we get non-uniform perfect zero-knowledge proofs for coNP. (Note that unlike $\mathsf{pr\text{-}SZK}$, $\mathsf{pr\text{-}PZK}$ is not known to be closed under complement.)

5.7.5 Application to problems with polynomial kernelizations

In this section we prove new limits to efficient compression for the Satisfiability problem on d-CNFs, and for some problems on graphs and hypergraphs, partially extending results of Dell and Van Melkebeek [DvM10] to handle two-sided error. First, we need some background.

Definition 5.7.8 (Hypergraphs, vertex covers, and cliques)**.** *For any integer $d \geq 2$, a d-*uniform hypergraph*, or d-*hypergraph*, is a set H of size-d subsets of a vertex set $V = [N]$. A* vertex cover *in a d-uniform hypergraph H is a subset of vertices that intersects all hyperedges in H. A subset $V' \subseteq V$ is a* clique *in H if every size-d subset of V' is a member of H.*

Clearly H has a vertex cover of size s exactly if the "complement" hypergraph $\overline{H} := \{e : |e| = d \ \wedge \ e \notin H\}$ contains a clique of size $N - s$.

Definition 5.7.9. *Define the parametrized problems*

$$d\text{-Vertex Cover} := \{\langle (H,s), 1^N\rangle : \ H \textit{ is a } d\textit{-hypergraph on } [N] \textit{ and contains a vertex cover of size } s\} ,$$

$$d\text{-Clique} \ := \ \{\langle (H,s), 1^N\rangle : \ H \textit{ is a } d\textit{-hypergraph on } [N] \textit{ and contains a clique of size } s\} .$$[32]

Also define the parametrized d-CNF Satisfiability problem

$$d\text{-SAT}_{par} \ := \ \{\langle \psi, 1^N\rangle : \ \psi \textit{ is a satisfiable } d\textit{-CNF on } N \textit{ variables}\} .$$

We will prove new limits on efficient compression for these problems with the help of the following powerful, ingenious reduction of Dell and Van Melkebeek.

Theorem 5.7.10 ([DvM10], Lemma 2)**.** *Fix $d \geq 2$, and let $T(n) : \mathbb{N}^+ \to \mathbb{N}^+$ be polynomially bounded. There is a deterministic polynomial-time OR-compression reduction*[33] *R^* for $L =$ 3-SAT,*[34] *with target language $L' = d$-*Clique*. For the first parameter we have $t_1(n) = T(n)$. The d-*Clique *instance $\langle (H,s), 1^N\rangle$ output by R^* satisfies*

$$N \ = \ O\left(n \cdot \max\left(n, T(n)^{1/d+o(1)}\right)\right) \ .$$

By straightforwardly combining Theorem 5.7.10 with our Theorem 5.7.4, we will prove the following theorem:

[32]This is a different parametrized problem than the two clique-based problems mentioned in Section 5.5.2.

[33](as in Definition 5.4.20)

[34]Here 3-SAT is just the usual language $\{\langle\psi\rangle : \psi \text{ is a satisfiable 3-CNF}\}$.

Theorem 5.7.11. *Let $d \geq 2, \varepsilon > 0$ be given. There is a $\beta = \beta(d, \varepsilon) > 0$ for which the following holds. Suppose that d-*Clique *has a polynomial-time compression reduction with output-size bound $O(N^{d-\varepsilon})$ and success probability $.5 + N^{-\beta}$; that is (in the terms of Definition 5.5.7), suppose that d-*Clique *is PPT-compressible with parameters c, ξ satisfying*

$$c(M, N, 1) \ \leq \ O(N^{d-\varepsilon}), \quad \xi(M, N, 1) \ \leq .5 - N^{-\beta} \ ,$$

with any target language L'.

Then, every language in NP is many-to-one reducible in non-uniform polynomial time to a problem in $\mathsf{pr\text{-}SZK}$ (and $\mathsf{NP} \subseteq \mathsf{coNP/poly}$).

*The same result holds if we replace d-*Clique *with d-*Vertex Cover *or d-*SAT$_{par}$.

Theorem 5.7.11 gives a version of [DvM10, Theorems 1 and 2] that applies to probabilistic reductions with two-sided error. However, our result does not apply to the more general setting of *oracle communication protocols*, to which those earlier results do apply (for co-nondeterministic protocols, and protocols avoiding false negatives).

Dell and Van Melkebeek use their techniques to show compression lower bounds for several other interesting graph problems (including the Feedback Vertex Set, Bounded-Degree Deletion, and Non-Planar Deletion problems) via reductions from 2-Vertex Cover [DvM10, Section 5.2]. Using our results and the reductions in [DvM10], one can also obtain similarly strong compression lower bounds for these problems for the two-sided error setting.

Proof of Theorem 5.7.11. We already described a simple reduction (in both directions) between the d-Vertex Cover and d-Clique problems that preserves the parameter N. Also, an instance of d-Vertex Cover on N vertices is efficiently reducible to a d-SAT instance over $O(N)$ variables [DvM10, Lemma 5]. Thus, it suffices to prove the result for d-Clique.

Let R be the compression reduction assumed to exist for d-Clique, with the value $\beta > 0$ to be determined later. Let $C > d$ be a large integer value, also to be determined.

We will define an OR-compression reduction R' for $L =$ 3-SAT and target language

L' from our assumption; this will allow us to apply Theorem 5.7.5. R' works as follows. We let $t_1(n) := n^C$. On input formulas $\psi_1, \ldots, \psi_{n^C}$, each of bit-length n, the reduction first computes $\langle (H, s), 1^N \rangle := R^*(\psi_1, \ldots, \psi_{n^C})$, where R^* is as in Theorem 5.7.10. Next, R' outputs the value $z := R(\langle (H, s), 1^N \rangle)$.

R' is clearly polynomial-time computable. To analyze R', fix length-n formulas $\psi_1, \ldots, \psi_{n^C}$, and let

$$b \ := \ \bigvee_{j=1}^{n^C} [\psi_j \in \text{3-SAT}] \ .$$

By the OR-compression property of the deterministic mapping R^*, we have

$$[b = 1] \iff \langle (H, s), 1^N \rangle \in d\text{-Clique} \ .$$

Then by the assumed reliability guarantee of R,

$$\begin{aligned} \Pr[L'(z) = b] \ &\geq \ .5 + N^{-\beta} \\ &\geq \ .5 + \left(O\left(n \cdot \max\left(n, n^{C/d + o(1)}\right)\right)\right)^{-\beta} \\ &\geq \ .5 + n^{-\beta(1 + C/d) + o(1)} \ . \end{aligned}$$

Thus the error bound $\xi(n)$ of our reduction R' is at most $.5 - n^{-\beta(1+C/d)+o(1)}$. Also, by the compression guarantee of R, the output z satisfies

$$\begin{aligned} |z| \ &\leq \ O(N^{d-\varepsilon}) \\ &\leq \ O\left(\left(n^{1+C/d+o(1)}\right)^{d-\varepsilon}\right) \\ &\leq \ O\left(n^{C-1+o(1)}\right) \ , \end{aligned}$$

with the last step valid provided we take $C > d(d+1)/\varepsilon$. Thus as an output-size bound for R', we may take $t_2(n) = O\left(n^{C-1+o(1)}\right)$. We evaluate

$$\begin{aligned} (1 - 2\xi(n))^2 - \sqrt{\frac{\ln 2}{2} \cdot \frac{t_2(n) + 1}{t_1(n)}} \ &\geq \ 4n^{-2(\beta(1+C/d) - o(1))} - O(n^{-.5+o(1)}) \\ &\geq \ n^{-\Omega(1)} \ , \end{aligned}$$

provided we take $\beta < .25(1 + C/d)^{-1}$. Thus under these settings, Eq. (5.17) holds. Then Theorem 5.7.5, item 1 gives the desired conclusion, since $L =$ 3-SAT is NP-complete. $\square$

5.8 Extension to quantum compression

In this section we will show that our results on OR- and AND-compression have analogues for the model in which the compression scheme is allowed to be a quantum algorithm, outputting a quantum state.

We assume familiarity with the basics of quantum computing and quantum information (for the needed background, consult [NC00]). However, readers without this background should be able to follow the overall structure of the argument if they are willing to regard "qubits," "quantum operations" "quantum algorithms," and "quantum measurements" as certain types of black-box objects, and accept some known facts about them. In particular, a "mixed state on m qubits" is a "quantum superposition" over classical m-bit strings. Let

$$\mathsf{MS}_m$$

denote the collection of m-qubit mixed states. (MS_m can be identified with the set of 2^m-by-2^m, trace-1, positive-semidefinite complex matrices.)

A "quantum operation" is a certain type of mapping $OP : \mathsf{MS}_m \to \mathsf{MS}_{m'}$, for some $m, m' > 0$. (The operations allowed by quantum physics are the *completely positive, trace-preserving (CPTP) maps*; these are a subset of the linear transformations mapping $\mathsf{MS}_m \subset \mathbb{C}^{m \times m}$ into $\mathsf{MS}_{m'} \subset \mathbb{C}^{m' \times m'}$.) We let

$$\mathsf{OP}_{m,m'}$$

denote the valid quantum operations from m-qubit into m'-qubit states.

"Quantum measurements" are measurements performed on quantum states to yield information about these states; in the quantum setting, measurements are inherently probabilistic, and alter the states being measured. See [NC00, Chapter 2] for a formal definition. Quantum states turn out to inherit some of the information-theoretic limitations of their classical counterparts; this fact will be the basis for our results on quantum compression.

5.8.1 Trace distance and distinguishability of quantum states

The *trace distance* is a metric on mixed quantum states from a shared state space [NC00]; we denote the trace distance between $\rho, \rho' \in \mathsf{MS}_m$ by $||\rho - \rho'||_{\mathrm{tr}} \in [0,1]$. Formally, treating ρ, ρ' as matrices,

$$||\rho - \rho'||_{\mathrm{tr}} \ := \ \frac{1}{2}\mathrm{Tr}\left[\sqrt{(\rho - \rho')^2}\right] \ .$$

This distance is intimately related to the distinguishability of quantum states. Suppose ρ, ρ' are two known states, and we are sent one or the other, each with equal probability (depending on the outcome of an unbiased coin flip $b \in \{0,1\}$). We want to guess b, by applying some series of quantum operations and measurements. For any ρ, ρ', it is known [NC00, Theorem 9.1] that our success probability at this task is maximized by using a single binary measurement,[35] depending on ρ, ρ', and that our maximum achievable success probability equals

$$\frac{1}{2}\left(1 + ||\rho - \rho'||_{\mathrm{tr}}\right) \ .$$

A probability distribution over mixed states is again a mixed state. Thus for a distribution $\mathcal{D}$ over a finite universe U and a mapping $R : U \to \mathsf{MS}_m$, $R(\mathcal{D})$ defines a quantum state. We use the following standard claim concerning such states, which follows from the distinguishability characterization of $|| \cdot ||_{\mathrm{tr}}$:

Claim 5.8.1. *For any distributions $\mathcal{D}, \mathcal{D}'$ over a shared finite universe U, and any*

[35](i.e., a measurement with two possible outcomes)

mapping $R : U \to \mathsf{MS}_m$, we have

$$||R(\mathcal{D}) - R(\mathcal{D}')||_{\mathrm{tr}} \;\leq\; ||\mathcal{D} - \mathcal{D}'||_{\mathrm{stat}} \;.$$

Similarly, for any valid quantum operation $OP \in \mathsf{OP}_{m,m'}$ and states $\rho, \rho' \in \mathsf{MS}_m$, we have

$$||OP(\rho) - OP(\rho')||_{\mathrm{tr}} \;\leq\; ||\rho - \rho'||_{\mathrm{tr}} \;.$$

5.8.2 Quantum f-compression

The following notion of quantum compression is modeled on Definition 5.4.20. The definition is made slightly more complicated by the fact that we no longer have the notion of a "target language" for our reduction; instead, we will require that the answer to our original instance of the decision problem $f \circ L$ be recoverable by some quantum measurement performed on the output state. (This measurement need not be efficiently performable, however.)

Definition 5.8.2 (Quantum f-compression reductions)**.** *Let L be a language, and let $f : \{0,1\}^* \to \{0,1\}$ be a Boolean function. Let $t_1(n), t_2(n) : \mathbb{N}^+ \to \mathbb{N}^+$ and $\xi(n) : \mathbb{N}^+ \to [0,1]$ be given.*

A quantum f-compression reduction for L, *with parameters $t_1(n), t_2(n), \xi(n)$, is a mapping $R(x^1, \ldots, x^m)$ outputting a mixed state ρ. There must also exist a family of (not necessarily efficiently-performable) binary quantum measurements $\{\mathcal{M}_n\}_{n>0}$ on $t_2(n)$-qubit states. We require the following properties: for all $(x^1, \ldots, x^{t_1(n)}) \in \{0,1\}^{t_1(n)\times n}$,*

1. *The state $\rho = R(x^1, \ldots, x^{t_1(n)})$ is on $t_2(n)$ qubits;*
2. *We have*

$$\Pr\left[\mathcal{M}_n(\rho) \;=\; f\left(L(x^1), \ldots, L(x^{t_1(n)})\right)\right] \;\geq\; 1 - \xi(n) \;.$$

If some reduction R as above is computable in quantum polynomial time, we say *that L is* QPT-f-compressible *with parameters* $(t_1(n), t_2(n), \xi(n))$.

5.8.3 Quantum complexity classes

We will be using the class $\mathsf{QIP}[k]$ of languages definable by *k-message, quantum interative proof systems* [Wat03]. Our treatment of these proof systems will be informal, since all the technical properties we need are summarized in theorems from prior work (for details see [Wat03, Wat02]). These are proof systems in which a computationally-unbounded Prover exchanges quantum messages with a quantum polynomial-time Verifier; a total of $k = k(n)$ messages are exchanged. Verifier sends the first message if k is even, or Prover if k is odd, and the parties alternate thereafter. We take $\mathsf{QIP} := \bigcup_{c>0} \mathsf{QIP}[n^c]$.

It was shown in [Wat03, KW00] that for any $3 \leq k(n) \leq \text{poly}(n)$, $\mathsf{PSPACE} \subseteq \mathsf{QIP}[k(n)] = \mathsf{QIP}[3]$; the latter class was recently shown to equal PSPACE [JJUW11]. Importantly for us, however, the class $\mathsf{QIP}[2]$ is not known to contain even coNP. The power of 3-message quantum proof systems is in contrast to the classical (private-coin) interactive-proof classes $\mathsf{IP}[k(n)]$, where for any constant $k \geq 2$, $\mathsf{IP}[k] = \mathsf{IP}[2] = \mathsf{AM}$, and the latter class is believed to be much weaker than $\mathsf{IP}[\text{poly}(n)] = \mathsf{PSPACE}$.

In what follows, we will actually find it more convenient to work with the promise-problem classes $\mathsf{pr\text{-}QIP}[k]$.[36] The results we've summarized carry over to the promise setting as well.

A model of quantum statistical zero-knowledge proofs was proposed by Watrous [Wat02], and used to define the class QSZK of promise problems having polynomial-time proof systems of this type.[37] We will use $\mathsf{pr\text{-}QSZK}$ to denote this class. Watrous showed in [Wat02] that Sahai and Vadhan's "statistical distance characterization" of $\mathsf{pr\text{-}SZK}$, embodied in Definition 5.4.13, has a quantum analogue. First, we need a promise problem involving trace distance. For a quantum circuit C with an m-qubit output

[36]This is to avoid having to define non-uniform versions of these classes, just as we avoided defining non-uniform versions of AM and SZK.

[37]Watrous's original model was of *honest-verifier* quantum statistical zero-knowledge proof systems; he later showed that these proof systems are equivalent in power to "cheating-verifier" ones [Wat09].

register, let ρ_C denote the output state of C on some fixed input state (say, the all-zeros state). We consider circuits built from a fixed, finite "universal" gate-set (see [NC00, Chapter 4]).

Definition 5.8.3. *For parameters $0 \le d \le D \le 1$, define the promise problem $\mathrm{TD}^{\ge D}_{\le d} = (\Pi_Y, \Pi_N)$ as follows:*

$$\Pi_Y \;:=\; \{\langle C, C'\rangle : ||\rho_C - \rho_{C'}||_{\mathrm{tr}} \ge D\} \ ,$$

$$\Pi_N \;:=\; \{\langle C, C'\rangle : ||\rho_C - \rho_{C'}||_{\mathrm{tr}} \le d\} \ .$$

In this definition, both $d = d(n)$ and $D = D(n)$ may be parameters depending on the input length $n = |\langle C, C'\rangle|$. (Here, the input description is a classical bit-string.)

Then, appealing to the result of [Wat02], we can use the following definition.

Definition 5.8.4. *Let* $\mathsf{pr\text{-}QSZK}$ *be defined as the class of promise problems for which there is a many-to-one (classical, deterministic) polynomial-time reduction from* Π *to* $\mathrm{TD}^{\ge 2/3}_{\le 1/3}$.

Theorem 5.8.5 ([Wat02])**.** $\mathsf{pr\text{-}QSZK}$ *is closed under complement.*

Theorem 5.8.6 ([Wat02])**.** $\mathsf{pr\text{-}QSZK} \subseteq \mathsf{pr\text{-}QIP}[2] \cap \mathsf{pr\text{-}coQIP}[2]$.

For upper bounds on the complexity of $\mathrm{TD}^{\ge D(n)}_{\le d(n)}$, we have the following two results, analogous to Theorems 5.4.15 and 5.4.14.

Theorem 5.8.7 (Follows from [Wat02])**.** *Suppose $0 \le d = d(n) < D = D(n) \le 1$ are polynomial-time computable, and satisfy $D > d + \frac{1}{\mathrm{poly}(n)}$. Then, $\mathrm{TD}^{\ge D}_{\le d} \in \mathsf{pr\text{-}QIP}[2]$.*

If we drop the requirement that d, D be computable, but keep the gap requirement, then $\mathrm{TD}^{\ge D}_{\le d}$ is many-to-one reducible in non-uniform (classical, deterministic) polynomial time to a problem in $\mathsf{pr\text{-}QIP}[2]$.

Theorem 5.8.7 follows from a "distinguishing protocol" analogous to that in Theorem 5.4.15.[38] Unlike the classical case, there is no known "non-uniform derandom-

[38]In [Wat02] only the case where D, d are constants is studied, but the result extends easily to when they are functions of n. Also, the second case, where we merely have the gap requirement, is not explicitly analyzed, but follows by a trivial modification of the proof of [Wat02, Theorem 4].

ization" result known for $\mathsf{QIP}[2]$ (or for other quantum classes). However, we do have a satisfying analogue of Theorem 5.4.14:

Theorem 5.8.8 (Follows from [Wat02])**.** *Suppose* $0 \le d = d(n) < D = D(n) \le 1$ *are polynomial-time computable, and satisfy* $D^2 > d + \frac{1}{\mathrm{poly}(n)}$. *Then,* $\mathrm{TD}^{\ge D}_{\le d} \in \mathsf{pr\text{-}QSZK}$.

If we drop the requirement that d, D *be computable, but keep the gap requirement, then* $\mathrm{TD}^{\ge D}_{\le d}$ *is many-to-one reducible in non-uniform (classical, deterministic) polynomial time to a problem in* $\mathsf{pr\text{-}QSZK}$.

5.8.4 Quantum distributional stability

We will use a quantum analogue of the distributional stability property:

Definition 5.8.9. *Let* $t, t', n \in \mathbb{N}^+$. *Given a mapping* $F : \{0,1\}^{t \times n} \to \mathsf{MS}_{t'}$, *and a collection* $\mathcal{D}_1, \ldots, \mathcal{D}_T$ *of mutually independent distributions over* $\{0,1\}^n$, *for* $j \in [t]$ *let*

$$\gamma_j \ := \ \mathop{\mathbb{E}}_{y \sim \mathcal{D}_j} \left[\left|\left| F\left(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, y, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t\right) - F\left(\mathcal{D}_1, \ldots, \mathcal{D}_t\right) \right|\right|_{\mathrm{tr}} \right] \ .$$

For $\delta \in [0,1]$, *say that* F *is* δ-quantumly-distributionally stable (or δ-QDS) *with respect to* $\mathcal{D}_1, \ldots, \mathcal{D}_t$ *if*

$$\frac{1}{t} \sum_{j=1}^{t} \gamma_j \ \le \ \delta \ .$$

The next lemma is analogous to Lemma 5.6.2.

Lemma 5.8.10. *Let* $t, t', n \in \mathbb{N}^+$. *Let* $R : \{0,1\}^{t \times n} \to \mathsf{MS}_{t'}$ *be given.*

Then, R *is* δ*-QDS with respect to any input distributions* $\mathcal{D}_1, \ldots, \mathcal{D}_t$, *where we may take either of the bounds*

1. $\delta := \sqrt{\frac{\ln 2}{2} \cdot \frac{t'}{t}}$;

2. $\delta := 1 - 2^{-\frac{t'}{t} - 2}$.

The slight improvement in the bounds comes from the fact that R outputs *exactly* t' qubits. The proof of Lemma 5.8.10 is very similar to that of Lemma 5.6.2, and is described in Section 5.11.

5.8.5 Building quantum disguising distributions

Next we prove quantum analogues of our Disguising-Distribution Lemmas. First, we have the following analogue of Lemma 5.6.3:

Lemma 5.8.11. *Let $t, t' \in \mathbb{N}^+$, and let $F(x^1, \ldots, x^T) : \{0,1\}^{T\times n} \to \mathsf{MS}_{t'}$ be given. Suppose F is δ-QDS with respect to input distribution $\mathcal{D}^{\otimes T}$, for every distribution $\mathcal{D}$ over $\{0,1\}^n$.*

Fix some distribution $\mathcal{D}$ over $\{0,1\}^n$, and let $x^1, \ldots, x^d$ be independently sampled from $\mathcal{D}$. Let $k^* \sim \mathcal{U}_{[d]}$. Let $\widehat{\mathcal{D}}$ denote the distribution defined by sampling uniformly from the multiset $\{x^k\}_{k\neq k^*}$. Define

$$\beta_j \ := \ \mathop{\mathbb{E}}_{k^*, x^1, \ldots, x^d} \left[\ \left|\left| R\left(\widehat{\mathcal{D}}^{\otimes(j-1)}, x^{k^*}, \widehat{\mathcal{D}}^{\otimes(T-j)}\right) \ - \ R\left(\widehat{\mathcal{D}}^{\otimes T}\right) \right|\right|_{\mathrm{tr}} \ \right] \ ,$$

where all the $\widehat{\mathcal{D}}$s are to be mutually independent (for fixed values of $x^1, \ldots, x^k$ and k^*). Then,

$$\frac{1}{T}\sum_{j=1}^{t} \beta_j \ \le \ \delta + 2T/d \ .$$

Proof. The proof is identical to that of Lemma 5.6.3, except that we replace statistical distance with trace distance[39] and appeal to Claim 5.8.1 to argue that applying R does not increase trace distance between states. □

After establishing a quantum analogue of Lemma 5.6.4, we have:

Lemma 5.8.12. *Suppose F obeys the assumptions of Lemma 5.8.11. Let $S \subseteq \{0,1\}^n$, and fix $d > 0$. Given any $\varepsilon > 0$, let $s := \lceil (.5\ln 2)n/\varepsilon^2 \rceil$. Then there exists a collection $K_1, \ldots, K_s$ of size-d multisets contained in S, such that for every $y \in S$ the following holds:*

$$\mathop{\mathbb{E}}_{a\sim\mathcal{U}_{[s]}, j^*\sim\mathcal{U}_{[t]}} \left[\ \left|\left| F\left(\mathcal{U}_{K_a}^{\otimes(j^*-1)}, y, \mathcal{U}_{K_a}^{\otimes(T-j^*)}\right) \ - \ F\left(\mathcal{U}_{K_a}^{\otimes T}\right) \right|\right|_{\mathrm{tr}} \ \right] \ \le \ \delta + 2T/(d+1) + \varepsilon \ .$$

[39](where appropriate—the input distributions we manipulate still are to be compared in statistical distance)

The proof is identical to that of Lemma 5.6.5, but again replacing statistical distance with trace distance. Then, by a proof analogous to that of Lemma 5.6.6, we obtain:

Lemma 5.8.13 (**Quantum Disguising-Distribution Lemma**). *Let $R(x^1, \ldots, x^t) : \{0,1\}^{t\times n} \to \mathsf{MS}_{t'}$ be any possibly-randomized mapping, where $n, t, t' \in \mathbb{N}^+$. Let $S \subseteq \{0,1\}^n$, and fix $d > 0$. Given any $\varepsilon > 0$, let $s := \lceil (.5\ln 2)n/\varepsilon^2 \rceil$. Let*

$$\widehat{\delta} := \min\left\{ \sqrt{\frac{\ln 2}{2}\cdot\frac{t'}{t}},\ \ 1 - 2^{-\frac{t'}{t}-2} \right\} .$$

Then there exists a collection $K_1, \ldots, K_s$ of size-d multisets contained in S, such that for every $y \in S$, we have

$$\mathop{\mathbb{E}}_{a\sim\mathcal{U}_{[s]},j^*\sim\mathcal{U}_{[t]}}\left[\left\| R\left(\mathcal{U}_{K_a}^{\otimes(j^*-1)},\ y,\ \mathcal{U}_{K_a}^{\otimes(t-j^*)}\right) \ - \ R\left(\mathcal{U}_{K_a}^{\otimes t}\right) \right\|_{\mathrm{tr}} \right] \\ \le\ \widehat{\delta} + 2t/(d+1) + \varepsilon \ .$$

5.8.6 Complexity upper bounds from quantum compression schemes

Now we are ready to prove a quantum analogue of Theorem 5.7.1.

Theorem 5.8.14. *Let L be any language. Suppose there is a QPT-*OR*-compression reduction $R(x^1, \ldots, x^t) : \{0,1\}^{t_1(n)\times n} \to \mathsf{MS}_{t_2(n)}$ for L with (not necessarily computable) parameters $t_1(n), t_2(n) : \mathbb{N}^+ \to \mathbb{N}^+$, and with error bound $\xi(n) < .5$. Let*

$$\widehat{\delta} := \min\left\{ \sqrt{\frac{\ln 2}{2}\cdot\frac{t'}{t}},\ \ 1 - 2^{-\frac{t'}{t}-2} \right\} .$$

1. If for some $c > 0$ we have

$$(1 - 2\xi(n)) - \widehat{\delta}\ \ge\ \frac{1}{n^c}\ , \tag{5.18}$$

then there is a non-uniform (classical, deterministic) polynomial-time many-to-

one reduction from L to a problem in $\mathsf{pr\text{-}QIP}[2]$.

2. *If we have the stronger bound*

$$(1 - 2\xi(n))^2 - \widehat{\delta} \ \geq \ \frac{1}{n^c} \ , \tag{5.19}$$

then L has a non-uniform (classical, deterministic) polynomial-time many-to-one reduction to a problem in $\mathsf{pr\text{-}QSZK}$.

Proof of Theorem 5.8.14. The proof is closely analogous to that of Theorem 5.7.1, except that our non-uniform reduction, on input y, outputs a description $\langle C, C' \rangle$ of a pair of *quantum* circuits. If $y \in L$, then $||\rho_C - \rho_{C'}||_{\mathrm{tr}} \geq D(n) := 1 - 2\xi(n)$; while if $y \notin L$, we have $||\rho_C - \rho_{C'}||_{\mathrm{tr}} \leq d(n) := 2\hbar - 1 + \frac{1}{n^c}$. Applying Theorems 5.8.7 and 5.8.8 gives us the complexity upper bounds in items 1 and 2. □

Using Theorem 5.8.14, we can prove quantum versions of Theorems 5.7.4 and 5.7.5, giving evidence against efficient quantum OR- and AND-compression for NP-complete languages, under the assumption that such languages are not non-uniformly reducible to problems in $\mathsf{pr\text{-}QIP}[2]$, or alternatively, in $\mathsf{pr\text{-}QSZK}$. A quantum analogue of Theorem 5.7.3, item 1 can be proved. We can also give an analogue of Theorem 5.7.7 regarding quantum compression for "expressive" parametrized problems. All of these quantum results treat compression reductions where the output state is of size determined by the various input parameters.

5.9 Alternative proofs of distributional stability

5.9.1 A proof based on Raz's lemma

R. Impagliazzo and S. Vadhan noted a similarity between distributional stability lemmas and a probabilistic lemma implicit in work of Raz [Raz98]. Vadhan pointed us to the following convenient form, given by Shaltiel in [Sha10, Lemma 3.1]:

Lemma 5.9.1. *There is a $c > 0$ for which the following holds. Let $X^1, \ldots, X^t$ be i.i.d. random variables, and T an event with $\Pr[T] \geq 2^{-\Delta}$. Then for the conditioned variables $X^j_{[T]}$ we have*

$$\mathbb{E}_{j \sim \mathcal{U}_{[t]}}[||X^j_{[T]} - X^j||_{\text{stat}}] \;\leq\; \varepsilon := \;\sqrt{\frac{c\Delta}{t}}\;.$$

With this lemma we can derive a distributional stability result as follows. Suppose $R : S^t \to \{0,1\}^{\leq t'}$ is given, and consider independent inputs $X^j \sim \mathcal{D}$ to R. We let $\mathbf{R}$ denote the random output value. For any output z of R, Lemma 5.9.1 above implies that

$$\mathbb{E}_{i \sim \mathcal{U}_{[n]}}[||X^j_{[\mathbf{R}=z]} - X^j||_{\text{stat}}] \;\leq\; \sqrt{c \log_2(1/\Pr[\mathbf{R}=z])/t}\;.$$

Taking expectations over $z \sim \mathbf{R}$ and using Jensen's inequality,

$$\begin{aligned}\mathbb{E}_{z \sim \mathbf{R}, i \sim \mathcal{U}_{[n]}}[||X^j_{[\mathbf{R}=z]} - X^j||_{\text{stat}}] \;&\leq\; \mathbb{E}_z\left[\sqrt{c \log_2(1/\Pr[\mathbf{R}=z])/t}\right] \\ &\leq\; \sqrt{\mathbb{E}_z\left[c \log_2(1/\Pr[\mathbf{R}=z])/t\right]} \\ &= \sqrt{c \cdot H_{rv}(t')/t} \\ &\text{(by the definition of Shannon entropy)} \\ &< \sqrt{c(t'+1)/t}\;.\end{aligned}$$

Let $\mathbf{R}' = R(Y^1, \ldots, Y^t)$ denote a sample of $\mathbf{R}$ based on inputs $Y^1, \ldots, Y^t \sim \mathcal{D}^{\otimes t}$ that are independent of $X^1, \ldots, X^t$. Now the crucial observation is that, for each $j \in [t]$, we have the chain of equalities

$$\mathbb{E}_z[||X^j_{[\mathbf{R}=z]} - X^j||_{\text{stat}}] \;=\; ||(X^j, \mathbf{R}) - (X^j, \mathbf{R}')||_{\text{stat}} \;=\; \mathbb{E}_{x^j \sim \mathcal{D}}[||\mathbf{R}_{[X^j = x^j]} - \mathbf{R}||_{\text{stat}}]\;.$$

Each equality follows from the "distinguishability interpretation" of statistical distance. Combining we get

$$\mathbb{E}_{x^j \sim \mathcal{D}}[||\mathbf{R}_{[X^j = x^j]} - \mathbf{R}||_{\text{stat}}] \;<\; \sqrt{c(t'+1)/t)}\;,$$

which is comparable to what we got from the previous approach (up to constant factors; here we have assumed i.i.d. variables X^j, but this is not essential for this approach).

5.9.2 A proof based on the Average Encoding Theorem

The Average Encoding Theorem of [KNTSZ07] is a tool in quantum information theory, that has the following classical analogue. (See [SV08, Fact 5], where a purely classical proof is given. We restate the result slightly, converting from ℓ^1 distance to statistical distance.)

Theorem 5.9.2. *Let X, M be random variables. Let Π be the distribution governing M. Then for the conditioned distributions $\Pi_{[X=x]}$ we have*

$$\sum_x \Pr[X = x] \cdot ||\Pi_{[X=x]} - \Pi||_{\mathrm{stat}} \;\le\; \sqrt{\frac{\ln 2 \cdot I(X;M)}{2}} \;.$$

Using Theorem 5.9.2 along with techniques suggested by Nayak and similar to those in the proof of [KNTSZ07, Theorem 5.4],[40] we can derive a distributional-stability result as follows. Again say we are given $R : S^t \to \{0,1\}^{t'}$, and consider independent inputs $X^j \sim \mathcal{D}$ to R, giving an output distribution denoted $\mathbf{R}$. Applying Theorem 5.9.2 to $X := X^j, M := \mathbf{R}$, we have

$$\mathbb{E}_{x^j \sim X^j}\left[||\mathbf{R}_{[X^j=x]} - \mathbf{R}||_{\mathrm{stat}}\right] \;\le\; \sqrt{\frac{\ln 2 \cdot I(X^j;\mathbf{R})}{2}} \;.$$

Averaging over $j \in [t]$ and applying Jensen's inequality and Lemma 5.4.4, we obtain

$$\begin{aligned}\mathbb{E}_{j \sim \mathcal{U}_{[t]}, x^j \sim X^j}\left[||\mathbf{R}_{[X^j=x]} - \mathbf{R}||_{\mathrm{stat}}\right] \;&\le\; \mathbb{E}_{j \sim \mathcal{U}_{[t]}}\left[\sqrt{\frac{\ln 2 \cdot I(X^j;\mathbf{R})}{2}}\right] \\ &\le\; \sqrt{\frac{\ln 2 \cdot \mathbb{E}_{j \sim \mathcal{U}_{[t]}}\left[I(X^j;\mathbf{R})\right]}{2}}\end{aligned}$$

[40](That proof uses a version of the Average Encoding Theorem that treats Hellinger distance rather than statistical distance.)

$$\leq \sqrt{\frac{\ln 2 \cdot (t'+1)}{2t}} \ .$$

5.10 Our original distributional stability lemma

In this section we include our original proof of a distributional stability lemma, based on coding-theoretic ideas. This lemma proves a distributional stability result for mappings $R : \{0,1\}^{t\times n} \to \{0,1\}^{\leq t'}$, where $t' + 2 \leq t$. In an earlier draft, we used more complicated ideas to prove complexity upper bounds from AND-compression reductions where $t' = O(t \log t)$. This latter case can now be handled in the same way as the case $t' \ll t$, using the alternative bound on distributional stability provided by Lemma 5.6.2, item 2.

First, we provide some further needed background.

5.10.1 Entropy and the unreliability of compressive encodings

It is a basic principle of information theory that one cannot reliably encode a uniformly-generated t-bit message by an encoding of length $t-2$ or less. (We can save essentially one bit by using a variable-length output.) Below we state and prove a standard claim that generalizes this fact, giving quantitative bounds on the reliability of compressive encoding methods.

Lemma 5.10.1. *Let $t \in \mathbb{N}^+$, and let U be some finite universe. Say we are given a possibly-randomized "encoding" function*

$$Enc(x, y) : \{0,1\}^t \times \{0,1\}^N \to U \ ,$$

depending on a "message" input $x \in \{0,1\}^t$ along with a "public randomness" input $y \in \{0,1\}^N$. (Enc may also have additional internal randomness.) Say we are also *given a (possibly-randomized, possibly-unreliable) "decoding" function*

$$Dec(x, y) : U \times \{0,1\}^N \to \{0,1\}^t \ ,$$

that also has access to the public randomness y.

Suppose X, Y are two independent random variables over $\{0,1\}^t, \{0,1\}^N$ respectively. For $j \in [t]$, let

$$p_j \ := \ \Pr_{X,Y}[Dec_j(Enc(X, Y), Y) \ = \ X_j] \ ,$$

where $X = (X_1, \ldots, X_t)$, and where Dec_j is the j^{th} output bit of Dec.

Let $p_{avg} := \frac{1}{t}\sum_{j=1}^{t} p_j$. Then, we must have

$$H(p_{avg}) \ \geq \ \frac{1}{t}\left(H_{rv}(X) - \log_2(|U|)\right) \ .$$

Our proof is closely modeled on the proof of a corresponding, but deeper, quantum result [KdW04, Appendix B].[41]. To prove Lemma 5.10.1, we will use another basic information-theoretic fact, Fano's inequality:

Lemma 5.10.2 (Fano). *[CT06, Chapter 2] Suppose Z_{in}, Z_{out} are two random variables: Z_{in} an "input message" over some alphabet Σ, and Z_{out} an "output message" over any domain. Let $\widetilde{Z}_{in}$ be a (possibly-randomized) function of Z_{out}, that attempts to recover the value Z_{in}. Let*

$$p_{err} \ := \ \Pr[\widetilde{Z}_{in} \neq Z_{in}] \ .$$

Here the randomness is over the entire experiment. Then, we have

$$H(p_{err}) \ + \ p_{err} \cdot \log_2\left(|\Sigma| - 1\right) \ \geq \ H_{rv}(Z_{in}|Z_{out}) \ .$$

We only use the case $|\Sigma| = 2$, so the second term on the left-hand side vanishes.

[41](or, Appendix A in the arxiv version. This part of [KdW04] is itself a rederivation of a result from [Nay99b]; a similar result and proof appears in Nayak's thesis [Nay99a, Theorem 3.2.8].)

Proof of Lemma 5.10.1. First we ask whether $p_{avg} \geq .5$. If not, we simply negate all of the decoding functions Dec_j, giving the modified average success probability $p'_{avg} = 1 - p_{avg}$, for which $H(p'_{avg}) = H(p_{avg})$. Next, note that the success probabilities p_j are taken over the randomness both in X and in Y (as well as in Enc, Dec). As Y is independent of X, we may non-uniformly fix some setting to Y that maximizes the sum of the conditional success probabilities. Then the re-modified average success probability satisfies $p''_{avg} \geq p'_{avg} \geq .5$. As $H(\cdot)$ is decreasing on $[.5, 1]$, any lower bound proved for $H(p''_{avg})$ will also lower-bound the $H(p_{avg})$ for the original encoding scheme with public randomness. Thus in the remainder of the proof, we assume $p_{avg} \geq .5$ and that the scheme uses no public randomness: our encoding Enc applies to X alone, and our decoding functions apply to the message $Enc(X)$ alone.

The chain rule for conditional entropy and the subadditivity of entropy imply that

$$H_{rv}(X|Enc(X)) \ = \ \sum_{j=1}^{t} H_{rv}(X_j|X_1, \ldots, X_{j-1}, Enc(X)) \ \leq \ \sum_{j=1}^{t} H_{rv}(X_j|Enc(X)) \ . \tag{5.20}$$

Next, we apply Fano's inequality, with $Z_{in} := X_j$, and $Z_{out} := Enc(X)$. Thus in this analysis we simply view $(X_{j'})_{j' \neq j}$ as additional sources of randomness in the encoding process. We let $\widetilde{Z}_{in} := Dec_j(Enc(X))$. X_j is binary—$\Sigma = \{0, 1\}$—so Fano's inequality gives

$$H(p_j) \ = \ H(1 - p_j) \ \geq \ H_{rv}(X_j|Enc(X)) \ .$$

Summing over j and using Eq. (5.20),

$$\begin{aligned} \sum_{j=1}^{t} H(p_j) \ &\geq \ H_{rv}(X|Enc(X)) \\ &\geq \ H_{rv}(X) - H_{rv}(Enc(X)) \ , \end{aligned} \tag{5.21}$$

again using subadditivity. Now, $Enc(X)$ is a message over U, so $H_{rv}(Enc(X)) \leq \log_2(|U|)$. Also, the function H is concave on $[0, 1]$. Applying these observations to

Eq. (5.21), and using Jensen's inequality, we have

$$H(p_{avg}) \;\geq\; \frac{1}{t}\sum_{j=1}^{t} H(p_j) \;\geq\; \frac{1}{t}\left(H_{rv}(X) - \log_2(|U|)\right) \ .$$

□

5.10.2 Bounds on the inverse entropy function

For $\alpha \in [0,1]$, we will denote by $H_+^{-1}(\alpha)$ the unique H-preimage of α in the range $[.5, 1]$. Similarly, let $H_-^{-1}(\alpha)$ denote the unique H-preimage of α in the range $[0, .5]$. The following bounds on the inverse entropy function are useful in understanding the bounds provided by our original distributional stability lemma (Lemma 5.10.4, to be presented shortly). These bounds on $H_+^{-1}(\cdot)$ are meant to be simple and illustrative, and are not quite best-possible.

Lemma 5.10.3. *We have the following facts:*

1. *If $m > 0$ is sufficiently large, then $H_+^{-1}(1/m) < 1 - 1/(4m\log_2 m)$.*
2. *$H_+^{-1}(1-\delta) \leq .5 + \frac{\sqrt{\ln 2}}{2}\sqrt{\delta} + O(\delta^{3/2})$.*

Proof. **(1.)** First consider any value $p \in (0, 1/2)$. We can upper-bound $H(p)$ in the following way:

$$\begin{aligned} H(p) \;&=\; p\log_2(1/p) \;+\; \underbrace{(1-p)}_{\leq 1}\log_2(\underbrace{1/(1-p)}_{\leq 1+2p}) \\ &<\; p\log_2(1/p) \;+\; \underbrace{2p}_{\text{(using }\log(1+c)<c\text{ for }c>0)} \\ &<\; 3p\log_2(1/p) \ . \end{aligned}$$

From this, one can easily verify that for $m \geq 10^6$, we have

$$H\left(\frac{1}{4m\log_2 m}\right) \;<\; \frac{1}{m} \ .$$

Thus for such m,

$$H_-^{-1}\left(\frac{1}{m}\right) \;>\; \frac{1}{4m\log_2 m}\ ,$$

so that

$$H_+^{-1}\left(\frac{1}{m}\right) \;=\; 1 - H_-^{-1}\left(\frac{1}{m}\right) \;<\; 1 - \frac{1}{4m\log_2 m}\ ,$$

giving item 1.

(2.) The binary entropy function H is infinitely differentiable on $(0,1)$, with

$$H(.5) = 1, \quad H'(.5) = 0, \quad H''(.5) = -4(\ln 2)^{-1}\ .$$

Thus for $\beta \in [0, .25)$ we have

$$H(.5+\beta) \;\le\; 1 - 4(\ln 2)^{-1}\beta^2 + O(\beta^3)\ .$$

By considering settings $\beta := \frac{\sqrt{\ln 2}}{2}\sqrt{\delta} \pm O(\delta^{3/2})$ and using that $H(\cdot)$ is decreasing on $(.5,1)$, we verify item 2. □

5.10.3 The lemma

Lemma 5.10.4. *Let $R(x^1,\ldots,x^t) : \{0,1\}^{t\times n} \to \{0,1\}^{\le t'}$ be any possibly-randomized mapping, where $t,t' \in \mathbb{N}^+$ satisfy $t'+2 \le t$.*

Then, R is δ-distributionally stable with respect to any input distributions $\mathcal{D}_1,\ldots,\mathcal{D}_t$, where

$$\delta \;:=\; 2H_+^{-1}\left(1 - \frac{t'+1}{t}\right) - 1\ .$$

We will prove Lemma 5.10.4 by a reduction to an encoding/decoding task that allows us to apply Lemma 5.10.1.

Proof of Lemma 5.10.4. Let $\mathcal{D}_1,\ldots,\mathcal{D}_t$ be independent distributions over $\{0,1\}^n$, and for $j \in [t]$, let γ_j be as in Definition 5.6.1 for $F := R$.

Consider the following encoding/decoding experiment involving t "Receivers." In the experiment, we will use R as a communication channel to attempt to transmit t

bits $b_1, \ldots, b_t$. Receiver $j \in [t]$ will be responsible for attempting to recover the value b_j. Formally:

1. For $j \in [t]$, let $y^j, w^j \sim \mathcal{D}_j$ (here y^j, w^j are independent of each other and of all other $y^{j'}, w^{j'}$);

2. Also, and independently, for $j \in [t]$ let $b_j \sim \mathcal{U}_{\{0,1\}}$;

3. If $b_j = 0$, let $x^j := y^j$. Otherwise, let $x^j := w^j$;

4. Let $z := R(x^1, \ldots, x^t)$ (a possibly-randomized value), and let z be sent to the Receivers. Let $\{y^j\}_{j \in [t]}$ be visible to the receivers as public randomness;

5. Each Receiver j outputs a guess $\tilde{b}_j$ for b_j, based on the values of the two random variables y^j and z). Specifically, Receiver j uses the maximum-likelihood rule $\tilde{b}_j := ML(b|y^j, z)$, described in Section 5.2.1.

Note that in making the guess $\tilde{b}_j$, Receiver j does *not* inspect the values $y^{j'}$, $j' \neq j$.

We analyze this experiment. First observe that, *conditioned* on a value y^j seen by Receiver j and on the value $b_j \in \{0, 1\}$ (which Receiver j does not see in the actual experiment), but leaving the other values $\{y^{j'}\}_{j \neq j}$ unconditioned, the conditional distribution on z is that $z \sim R(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, y^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t)$ if $b_j = 0$, and $z \sim R(\mathcal{D}_1, \ldots, \mathcal{D}_j, \ldots, \mathcal{D}_t)$ if $b_j = 1$.

Also, b_j is unbiased and independent of y^j. Thus, by the distinguishability interpretation of statistical distance (see Section 5.2.1), Receiver j's success probability in guessing b_j, conditioned exclusively on an observed value y^j, equals

$$\frac{1}{2}\left(1 + \left|\left|R(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, y^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t) - R(\mathcal{D}_1, \ldots, \mathcal{D}_j, \ldots, \mathcal{D}_t)\right|\right|_{\text{stat}}\right) .$$

Thus Receiver j's *overall* success probability in the experiment is precisely $\frac{1}{2}(1 + \gamma_j)$, where γ_j is as in the definition of distributional stability for R with respect to $\mathcal{D}_1, \ldots, \mathcal{D}_t$.

In our present setup, we can regard

$$z \ = \ R(x^1, \ldots, x^t) \ =: \ Enc(b_1, \ldots, b_t, y^1, \ldots, y^t)$$

as a randomized encoding function of $b_1, \ldots, b_t$, with public randomness $y^1, \ldots, y^t$ and additional private randomness $w^1, \ldots, w^t$. Similarly, we can view

$$(\tilde{b}_1, \ldots, \tilde{b}_t) \; =: \; Dec(z, y^1, \ldots, y^t)$$

as a (deterministic) decoding function. The success probability of our encoding/decoding experiment in successfully decoding b_j is $\frac{1}{2}(1 + \gamma_j)$.

Now $H_{rv}(b_1, \ldots, b_t) = t$, and $H_{rv}(Enc(b_1, \ldots, b_t, y^1, \ldots, y^t)) \leq \log_2 \left(\left| \{0,1\}^{\leq t'} \right| \right) < t' + 1$. Applying Lemma 5.10.1, we find that

$$H\left(\frac{1}{2}\left(1 + \frac{1}{t}\sum_{j=1}^{t} \gamma_j\right)\right) \geq \frac{1}{t}\left(t - (t'+1)\right) \; = \; 1 - \frac{t'+1}{t} \; ,$$

which implies that

$$\frac{1}{t}\sum_{j=1}^{t} \gamma_j \; \leq 2H_+^{-1}\left(1 - \frac{t'+1}{t}\right) - 1 \; = \delta \; .$$

Thus, R is δ-DS with respect to $\mathcal{D}_1, \ldots, \mathcal{D}_t$. As these distributions were arbitrary, this proves Lemma 5.10.4. □

It is also possible to give a slightly shorter and more direct proof of Lemma 5.10.4, using the same basic information-theoretic steps from the proof of Lemma 5.10.1; one can apply these to steps 1-4 of the experiment described above, without defining the decoding process.[42] However, we feel that the reduction to encoding/decoding is more intuitive, and more clearly illustrates a general idea that has been useful in various settings, e.g., in [Reg11].

5.11 Proof of quantum distributional stability

Here we describe how Lemma 5.8.10 is proved. We use various concepts and results of quantum information theory. In particular, we assume familiarity with the notion of

[42] I thank James Lee and Avi Wigderson for suggesting to find a more direct information-theoretic proof.

bipartite and reduced states. We let $S(\rho) := -\mathrm{Tr}(\rho \log_2 \rho)$ denote the *Von Neumann entropy* of a quantum state (here, identifying ρ with its density matrix). In analogy to the classical case, the entropy of a d-qubit state can be shown to be at least 0 and at most d. For a bipartite state $\rho_{A,B}$ on subsystems A, B, we let ρ_A (resp. ρ_B) denote the reduced state over A (resp. B)

We define the *quantum mutual information* between subsystems A, B of a bipartite state ρ_{AB} as

$$I_q(A;B) \ := \ S(\rho_A) + S(\rho_B) - S(\rho_{AB}) \ .$$

See [NC00], and Chapter 11 in particular, for more background in quantum information.

We will sometimes speak of quantum systems containing a subsystem that is a classical random variable X. By this we mean a state of form

$$\rho_{X,Y} \ = \sum_{x \in \mathrm{supp}(X)} \Pr[X = x] \cdot |x\rangle\langle x| \otimes \sigma_x \ , \tag{5.22}$$

for some collection of quantum states $\{\sigma_x\}$ on a fixed number of qubits (the "Y-subsystem").

Lemma 5.11.1. *[NC00, Theorem 11.8.5, p. 513] For a classical random variable X, and a state of the form in Eq. (5.22), we have*

$$S(\rho_{X,Y}) \ = \ H(X) + \sum_x \Pr[X = x] S(\sigma_x) \ .$$

In particular, considering the case where Y is an empty register, we have $S(\rho_X) = H(X)$.

We have the following elementary bound on the quantum mutual information between a classical message and its quantum encoding.[43]

Lemma 5.11.2. *For a classical random variable X, and a state of the form in*

[43] I thank Scott Aaronson and Thomas Vidick for helping me to understand this fact.

Eq. (5.22), with the states $\{\sigma_x\}$ on d qubits, we have

$$I_q(X:Y) \;=\; S(\rho_Y) - \sum_x \Pr[X=x]S(\sigma_x) \;\leq\; d\;.$$

Proof. Using Lemma 5.11.1, we calculate

$$\begin{aligned} I_q(X:Y) &= S(\rho_X) + S(\rho_Y) - S(\rho_{XY}) \\ &= H(X) + S(\rho_Y) - \left[H(X) + \sum_x \Pr[X=x]S(\sigma_x)\right] \\ &= S(\rho_Y) - \sum_x \Pr[X=x]S(\sigma_x) \\ &\leq d\;, \end{aligned}$$

since ρ_Y consists of d qubits and $S(\sigma_x) \geq 0$ for each x. □

Not all properties of classical entropy and mutual information are inherited by their quantum counterparts.[44] However, we have [Nay99a, p. 33 and Appendix A]:

Fact 5.11.3. *Quantum mutual information obeys the following properties, for all X, Y, Z:*

1. $I_q(X;Y) = I_q(Y;X) \geq 0$;
2. $I_q(X;(Y,Z)) = I_q(X;Y) + I_q((X,Y);Z) - I_q(Y;Z)$;
3. *(Strong subadditivity)* $I_q(X;(Y,Z)) \geq I_q(X;Y)$;
4. $I_q(X;Z) = 0$ *if the subsystems X, Z are independent classical random variables.*

Item 3 is a nontrivial fact in the quantum setting, with multiple equivalent formulations; see [NC00, Chapter 11].

With these facts in hand, the proof of Lemma 5.11.4 below exactly follows that of Lemma 5.4.4.

[44]For example, it is not generally true that $S(\rho_{AB}) \leq S(\rho_A)$ by analogy with the fact that $H_{rv}((X,Y)) \geq H_{rv}(X)$. Fact 5.4.1. Note, though, that we don't use this classical fact in proving Lemma 5.6.2.

Lemma 5.11.4. *If $X^1, \ldots, X^t$ are independent classical random variables and Y a quantum subsystem, then*

$$I_q(Y; (X^1, \ldots, X^t)) \;\geq\; \sum_{j \in [t]} I_q(Y; X^j) \; .$$

Next, we need quantum analogues of Pinsker's and Vajda's inequalities. For mixed states ρ, σ over the same number of qubits, define the *relative entropy* (a quantum analogue of Kullback-Leibler divergence) as

$$S(\rho || \sigma) \;:=\; \mathrm{Tr}(\rho \log_2(\sigma)) - S(\rho) \; .$$

We also have the following analogue of Fact 5.4.6 [KNTSZ07, p. 10]:

Fact 5.11.5. $I(A; B) = S(\rho_{AB} || \rho_A \otimes \rho_B)$.

A quantum Pinsker inequality was explicitly proved in [KNTSZ07, Theorem III.1].[45] However, that proof actually demonstrates a more general principle:

Theorem 5.11.6 ([KNTSZ07]). *Suppose that for some $\alpha, \beta \geq 0$, the (classical) statistical distance and Kullback-Leibler divergence obey the relationship*

$$||X - Y||_{\text{stat}} \;\geq\; \alpha \quad \Longrightarrow \quad D_{\text{KL}}(X || Y) \;\geq\; \beta \; .$$

for every pair of classical distributions X, Y.

Then, for any pair ρ, σ of quantum states,

$$||\rho - \sigma||_{\text{tr}} \;\geq\; \alpha \quad \Longrightarrow \quad S(\rho || \sigma) \;\geq\; \beta \; .$$

Combining this principle with the classical Pinsker and Vajda inequalities, we obtain:

[45] An earlier version appears in [OP04].

Corollary 5.11.7 (Quantum Pinsker inequality)**.** *For any states ρ, σ,*

$$S(\rho||\sigma) \ \geq \ \frac{2}{\ln 2} \cdot ||\rho - \rho'||_{\mathrm{tr}}^2$$

Corollary 5.11.8 (Quantum Vajda inequality)**.** *For any states ρ, σ,*

$$S(\rho||\sigma) \ \geq \ \frac{1}{\ln 2}\left(\ln\left(\frac{1}{1 - ||\rho - \sigma||_{\mathrm{tr}}}\right) - 1\right).$$

In the quantum setting we let $\mathbf{R}$ denote the mixed quantum state $R(X^1, \ldots, X^t)$, where $X^j \sim \mathcal{D}_j$. The inequality

$$I_q((X^1, \ldots, X^t); \mathbf{R}) \ \leq \ t'$$

follows from Lemma 5.11.1, since $\mathbf{R} \in \mathsf{MS}_{t'}$. With the assembled tools in hand, the proof of Lemma 5.8.10 is essentially identical to that of Lemma 5.6.2. The one difference is that the classical equality

$$||(X^j, \mathbf{R}) - (Y^j, \mathbf{R})||_{\mathrm{stat}} \ = \ \mathbb{E}_{x^j \sim \mathcal{D}_j}\left[\left|\left|R\left(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, x^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t\right) - R\left(\mathcal{D}_1, \ldots, \mathcal{D}_t\right)\right|\right|_{\mathrm{stat}}\right]$$

we used there is replaced by the *in*equality

$$||(X^j, \mathbf{R}) - (Y^j \otimes \mathbf{R})||_{\mathrm{tr}} \ \geq \ \mathbb{E}_{x^j \sim \mathcal{D}_j}\left[\left|\left|R\left(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, x^j, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t\right) - R\left(\mathcal{D}_1, \ldots, \mathcal{D}_t\right)\right|\right|_{\mathrm{tr}}\right] .$$

This inequality follows by considering the experiment that first measures the X^j register, then performs an optimal distinguishing measurement on $\mathbf{R}$ conditioned on the outcome of the first measurement. Note that this inequality goes in the needed direction.

5.12 Questions for further study

1. Can we extend the limitations we show on efficient compression for AND(SAT) and OR(SAT), to give corresponding lower bounds on the cost of solving these

problems in the *oracle communication model* studied by Dell and Van Melkebeek [DvM10]? These authors were able to extend the lower bounds of [FS11] for OR(SAT) to this more general setting. Proceeding by straightforward analogy in our case seems to fail, however.

2. Using our results on the infeasibility of compression for AND(SAT), can we extend the work of [DvM10] to prove new kernel-size lower bounds for interesting problems *with* polynomial kernels, under the assumption $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$?

3. Can we obtain a better quantitative understanding of the limits to efficient f-compression of NP-complete languages, where f is a combining function other than OR or AND? The case $f = \bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{m} x^{i,j} \right)$ is an interesting candidate for study.

NOTE: The last 2 paragraphs of Question 3 are incorrect due to the mistake in Sec. 5.7.3. However, the open problem about the Index function below is solved by the methods of our revised ECCC paper.

   As noted in Section 5.1.3, our methods imply infeasibility of strong f-compression to any target language, for any *monotone* combining f depending on all variables. We can also handle non-monotone f, provided the *sensitivity* $s(f)$ satisfies $s(f) = n^{\Omega(1)}$. (See [BdW02] for background on sensitivity.)

   If f is non-monotone and depends on all variables, but has low sensitivity, we seem to need the additional requirement that the target language L' be in NP in order to prove compression lower bounds (under the assumption $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$). As a concrete example of a question this leaves open: is there a strong f-compression reduction from SAT to *any* target language L' (beyond NP), when f is the "index function"

$$f(x_1, \ldots, x_{\log n}, y_1, \ldots, y_n) \ := \ y_x \ ?$$

4. Can we find other applications for the Disguising-Distribution Lemma?

5.13 Chapter acknowledgments

I thank Scott Aaronson, Hans Bodlaender, Holger Dell, Lance Fortnow, Russell Impagliazzo, James Lee, Dieter van Melkebeek, Ashwin Nayak, Karolina Soltys, Salil Vadhan, Thomas Vidick, Avi Wigderson, Ryan Williams, and several anonymous reviewers for helpful comments. Thanks especially to Russell, Ashwin, and Salil for allowing me to include their alternative proof suggestions.

Bibliography

[Aar05] Scott Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1(1):1–28, 2005. Earlier version in CCC '04.

[AB87] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[ABG$^+$01] Andris Ambainis, Harry Buhrman, William I. Gasarch, Bala Kalyanasundaram, and Leen Torenvliet. The communication complexity of enumeration, elimination, and selection. *J. Comput. Syst. Sci.*, 63(2):148–185, 2001. Earlier version in CCC '00.

[Adl78] Leonard M. Adleman. Two theorems on random polynomial time. In *19th IEEE FOCS*, pages 75–83, 1978.

[AFW81] L. Auslander, E. Feig, and S. Winograd. Direct sums of bilinear algorithms. *Linear Algebra and its Applications*, 38:175 – 192, 1981.

[AH91] William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *J. Comput. Syst. Sci.*, 42(3):327–345, 1991.

[AHPV05] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets, 2005. Survey article.

[AKW90] Noga Alon, Mauricio Karchmer, and Avi Wigderson. Linear circuits over GF(2). *SIAM J. Comput.*, 19(6):1064–1067, 1990.

[Alt94] Ingo Althöfer. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra and its Applications*, 199, Supplement 1(0):339 – 355, 1994.

[AMRR11] Andris Ambainis, Loïck Magnin, Martin Roetteler, and Jérémie Roland. Symmetry-assisted adversaries for quantum state generation. In *IEEE Conference on Computational Complexity*, pages 167–177, 2011.

[And87] A. E. Andreev. A method for obtaining efficient lower bounds for monotone complexity. *Algebra and Logic*, 26:1–18, 1987.

[AP94] Noga Alon and Pavel Pudlák. Superconcentrators of depths 2 and 3; odd levels help (rarely). *J. Comput. Syst. Sci.*, 48(1):194–202, 1994.

[ASdW09] Andris Ambainis, Robert Spalek, and Ronald de Wolf. A new quantum lower bound method, with applications to direct product theorems and time-space tradeoffs. *Algorithmica*, 55(3):422–461, 2009. Earlier version in STOC '06.

[Aus06] David Austin. How Google finds your needle in the web's haystack (AMS feature column, online), December 2006. http://www.ams.org/samplings/feature-column/fcarc-pagerank.

[BBCR10] Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. In *42nd ACM STOC*, pages 67–76, 2010.

[BDFH09] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. Earlier version in ICALP '08.

[BDKW10] Amos Beimel, Sebastian Ben Daniel, Eyal Kushilevitz, and Enav Weinreb. Choosing, agreeing, and eliminating in communication complexity. In *37th ICALP*, pages 451–462, 2010.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[BFP$^+$73] Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.

[BG81] Charles H. Bennett and John Gill. Relative to a random oracle A, P$^{\mathrm{A}}$!= NP$^{\mathrm{A}}$!= co-NP$^{\mathrm{A}}$ with probability 1. *SIAM J. Comput.*, 10(1):96–113, 1981.

[BH08] Harry Buhrman and John M. Hitchcock. NP-hard sets are exponentially dense unless coNP $\subseteq$ NP/poly. In *IEEE Conference on Computational Complexity*, pages 1–7, 2008.

[BI87] Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes (extended abstract). In *FOCS*, pages 118–126, 1987.

[Bir67] Garrett Birkhoff. *Lattice Theory.* American Mathematical Society, 1967.

[BJK11a] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *STACS*, pages 165–176, 2011.

[BJK11b] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernel bounds for path and cycle problems. In *IPEC*, pages 145–158, 2011.

[BJK11c] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Preprocessing for treewidth: A combinatorial analysis through kernelization. In *38th ICALP*, pages 437–448, 2011.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984. Earlier version in FOCS '82.

[BNRdW07] Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory Comput. Syst.*, 40(4):379–395, 2007. Earlier version in STACS '05.

[BR11] Mark Braverman and Anup Rao. Information equals amortized communication. In *52nd IEEE FOCS*, pages 748–757, 2011.

[Bra12] Mark Braverman. Interactive information complexity. In *44th ACM STOC*, pages 505–524, 2012.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theor. Comput. Sci.*, 22:317–330, 1983.

[Bsh89] Nader H. Bshouty. On the extended direct sum conjecture. In *21st ACM STOC*, pages 177–185, 1989.

[Bsh98] Nader H. Bshouty. On the direct sum conjecture in the straight line model. *J. Complexity*, 14(1):49–62, 1998. Earlier version in ESA '93.

[BTY11] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. Earlier version in ESA '09.

[Bub86] Siegfried Bublitz. Decomposition of graphs and monotone formula size of homogeneous functions. *Acta Informatica*, 23(6):689–696, 1986.

[Buh] Harry Buhrman. Personal communication.

[CCDF97] Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119 – 138, 1997.

[CDH$^+$00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In *EUROCRYPT*, pages 453–469, 2000.

[CFL83] Ashok K. Chandra, Steven Fortune, and Richard J. Lipton. Lower bounds for constant depth circuits for prefix problems. In *10th ICALP*, pages 109–117, 1983.

[CFL85] Ashok K. Chandra, Steven Fortune, and Richard J. Lipton. Unbounded fan-in circuits and associative functions. *J. Comput. Syst. Sci.*, 30(2):222–234, 1985.

[CFM11] Yijia Chen, Jörg Flum, and Moritz Müller. Lower bounds for kernelizations and other preprocessing procedures. *Theory Comput. Syst.*, 48(4):803–839, 2011. Earlier version in CiE '09.

[CGS12] Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic applications of Baur-Strassen's theorem: Shortest cycles, diameter and matchings. *CoRR*, abs/1204.1616, 2012.

[Che08a] Dmitriy Yu. Cherukhin. Lower bounds for Boolean circuits with finite depth and arbitrary gates. *Electronic Colloquium on Computational Complexity (ECCC)*, TR08-032, 2008.

[Che08b] Dmitriy Yu. Cherukhin. Lower bounds for depth-2 and depth-3 Boolean circuits with arbitrary gates. In *3rd CSR*, pages 122–133, 2008.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[com10] comScore. comscore reports global search market growth of 46 percent in 2009 (web article), January 2010. http://www.comscore.com/Press_Events/Press_Releases/2010/1/Global_Search_Market_Grows_46_Percent_in_2009.

[CRR90] Leonard S. Charlap, Howard D. Rees, and David P. Robbins. The asymptotic probability that a random biased matrix is invertible. *Discrete Mathematics*, 82(2):153–163, 1990.

[CSWY01] Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *42nd IEEE FOCS*, pages 270–278, 2001.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[DDPW83] Danny Dolev, Cynthia Dwork, Nicholas Pippenger, and Avi Wigderson. Superconcentrators, generalizers and generalized connectors with limited depth (preliminary version). In *15th ACM STOC*, pages 42–51, 1983.

[DF99] R. G. Downey and M.R. Fellows. *Parametrized Complexity*. Springer (Monographs in Computer Science), 1st edition, 1999.

[DLS09] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In *36th ICALP*, pages 378–389, 2009.

[DM12] Holger Dell and Dániel Marx. Kernelization of packing problems. In *23rd ACM-SIAM SODA*, pages 68–81, 2012.

[DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms.* Cambridge University Press, 1st edition, 2009.

[Dru12] Andrew Drucker. Improved direct product theorems for randomized query complexity. *Computational Complexity*, 21(2):197–244, 2012.

[DvM10] Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In *42nd ACM STOC*, pages 251–260, 2010.

[FGM$^+$89] Martin Fürer, Oded Goldreich, Yishay Mansour, Michael Sipser, and Stathis Zachos. On completeness and soundness in interactive proof systems. *Advances in Computing Research*, 5:429–442, 1989.

[FHT03] Alexei A. Fedotov, Peter Harremoës, and Flemming Topsøe. Refinements of Pinsker's inequality. *IEEE Transactions on Information Theory*, 49(6):1491–1498, 2003.

[FKNN95] Tomás Feder, Eyal Kushilevitz, Moni Naor, and Noam Nisan. Amortized communication complexity. *SIAM J. Comput.*, 24(4):736–750, 1995. Earlier version in FOCS '91.

[For87] Lance Fortnow. The complexity of perfect zero-knowledge (extended abstract). In *19th ACM STOC*, pages 204–209, 1987.

[For89] Lance Fortnow. Complexity-theoretic aspects of interactive proof systems, 1989. Ph.D. thesis, MIT LCS.

[FP11] Franz Franchetti and Markus Püschel. FFT (Fast Fourier Transform). In David A. Padua, editor, *Encyclopedia of Parallel Computing*, pages 658–671. Springer, 2011.

[FRPU94] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, 1994. Earlier version in STOC '90.

[FS11] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. Earlier version in STOC '08.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. Earlier version in FOCS '81.

[FW84] Ephraim Feig and Shmuel Winograd. On the direct sum conjecture. *Linear Algebra and its Applications*, 63:193 – 219, 1984.

[FZ77] Charles M. Fiduccia and Yechezkel Zalcstein. Algebras having linear multiplicative complexities. *J. ACM*, 24(2):311–331, 1977.

[GG81] M. J. Fischer Giulia Galbiati. On the complexity of 2-output boolean networks. *Theor. Comput. Sci.*, 16:177–185, 1981.

[GHK$^+$12] Anna Gál, Kristoffer Arnsfelt Hansen, Michal Koucký, Pavel Pudlák, and Emanuele Viola. Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates. In *44th ACM STOC*, pages 479–494, 2012.

[GN07] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, TR95-050, 1995.

[Gol10] Oded Goldreich. A brief introduction to property testing. In Oded Goldreich, editor, *Property Testing*, volume 6390 of *Lecture Notes in Computer Science*, pages 1–5. Springer, 2010.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *18th ACM STOC*, pages 59–68, 1986.

[GSV98] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *30th ACM STOC*, pages 399–408, 1998.

[GV11] Oded Goldreich and Salil P. Vadhan. On the complexity of computational problems regarding distributions (a survey). *Electronic Colloquium on Computational Complexity (ECCC)*, TR11-004:4, 2011.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *18th ACM STOC*, pages 6–20, 1986.

[HH91] Juris Hartmanis and Lane A. Hemachandra. One-way functions and the nonisomorphism of NP-complete sets. *Theor. Comput. Sci.*, 81(1):155–163, 1991.

[HJMR10] Prahladh Harsha, Rahul Jain, David A. McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. *IEEE Transactions on Information Theory*, 56(1):438–449, 2010. Earlier version in CCC '07.

[HN10] Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010. Earlier version in FOCS '06.

[HS11] Thomas Holenstein and Grant Schoenebeck. General hardness amplification of predicates and puzzles - (extended abstract). In *8th TCC*, pages 19–36, 2011.

[HW12] Danny Hermelin and Xi Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *23rd ACM-SIAM SODA*, pages 104–113, 2012.

[IJKW10] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010. Earlier version in STOC '08.

[IK10] Russell Impagliazzo and Valentine Kabanets. Constructive proofs of concentration bounds. In *14th RANDOM*, pages 617–631, 2010.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th IEEE FOCS*, pages 538–545, 1995.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. Size-depth tradeoffs for threshold circuits. *SIAM J. Comput.*, 26(3):693–707, 1997. Earlier version in STOC '93.

[IRW94] Russell Impagliazzo, Ran Raz, and Avi Wigderson. A direct product theorem. In *IEEE Structure in Complexity Theory Conference*, pages 88–96, 1994.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *29th ACM STOC*, pages 220–229, 1997.

[Jai10a] Rahul Jain. Strong direct product conjecture holds for all relations in public coin randomized one-way communication complexity. *CoRR*, abs/1010.0522, 2010.

[Jai10b] Rahul Jain. A strong direct product theorem for two-way public coin communication complexity. *CoRR*, abs/1010.0846, 2010.

[JáJ92] Joseph JáJá. *An Introduction to Parallel Algorithms.* Addison-Wesley, 1992.

[JJUW11] Rahul Jain, Zhengfeng Ji, Sarvagya Upadhyay, and John Watrous. QIP = PSPACE. *J. ACM*, 58(6):30, 2011. Earlier version in STOC '10.

[JKS10] Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Inf. Process. Lett.*, 110:893–897, 2010.

[JPY12] Rahul Jain, Attila Pereszlényi, and Penghui Yao. A direct product theorem for bounded-round public-coin randomized communication complexity. *CoRR*, abs/1201.1666, 2012.

[JRS03] Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. A direct sum theorem in communication complexity via message compression. In *30th ICALP*, pages 300–315, 2003.

[JS10] Stasys Jukna and Georg Schnitger. Circuits with arbitrary gates for random operators. *CoRR*, abs/1004.5236, 2010.

[JT86] Joseph JáJá and Jean Takche. On the validity of the direct sum conjecture. *SIAM J. Comput.*, 15(4):1004–1020, 1986.

[Juk10a] Stasys Jukna. Entropy of operators or why matrix multiplication is hard for depth-two circuits. *Theory Comput. Syst.*, 46(2):301–310, 2010.

[Juk10b] Stasys Jukna. Representing (0, 1)-matrices by Boolean circuits. *Discrete Mathematics*, 310(1):184–187, 2010.

[Juk12] Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers.* Algorithms and Combinatorics Series, #27. Springer-Verlag New York, LLC, 2012.

[KdW04] Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *J. Comput. Syst. Sci.*, 69(3):395–420, 2004. Earlier version in STOC '03.

[Kla10] Hartmut Klauck. A strong direct product theorem for disjointness. In *42nd ACM STOC*, pages 77–86, 2010.

[KN96] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1996.

[KNTSZ07] Hartmut Klauck, Ashwin Nayak, Amnon Ta-Shma, and David Zuckerman. Interaction in quantum communication. *IEEE Transactions on Information Theory*, 53(6):1970–1982, 2007.

[Knu73] Donald E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching.* Addison-Wesley, 1973.

[Kor03] A. D. Korshunov. Monotone Boolean functions. *Russian Math. Surveys*, 58:929–1001, 2003.

[Kra12] Stefan Kratsch. Co-nondeterminism in compositions: a kernelization lower bound for a Ramsey-type problem. In *23rd ACM-SIAM SODA*, pages 114–122, 2012.

[KRW95] Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3/4):191–204, 1995. Earlier version in *Structure in Complexity Theory Conference '91.*

[KŠdW07] Hartmut Klauck, Robert Špalek, and Ronald de Wolf. Quantum and classical strong direct product theorems and optimal time-space tradeoffs. *SIAM J. Comput.*, 36(5):1472–1493, 2007. Earlier version in FOCS '04.

[KW00] Alexei Kitaev and John Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. In *32nd ACM STOC*, pages 608–617, 2000.

[KW12] Stefan Kratsch and Magnus Wahlström. Compression via matroids: a randomized polynomial kernel for odd cycle transversal. In *23rd ACM-SIAM SODA*, pages 94–103, 2012.

[Lan12] J. M. Landsberg. *Tensors: Geometry and Applications*. Graduate Studies in Mathematics, vol. 128. American Mathematical Society, 2012.

[LMM03] Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Playing large games using simple strategies. In *4th ACM Conference on Electronic Commerce*, pages 36–41, 2003.

[Lok09] Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1-2):1–155, 2009.

[LR12] Troy Lee and Jérémie Roland. A strong direct product theorem for quantum query complexity. In *IEEE Conference on Computational Complexity*, pages 236–246, 2012.

[LSŠ08] Troy Lee, Adi Shraibman, and Robert Špalek. A direct product theorem for discrepancy. In *IEEE Conference on Computational Complexity*, pages 71–80, 2008.

[Lup56] O.B. Lupanov. On rectifier and switching-and-rectifier schemes. *Dokl. Akad. Nauk SSSR*, 111:1171–1174, 1956.

[LY94] Richard J. Lipton and Neal E. Young. Simple strategies for large zero-sum games with applications to complexity theory. In *26th ACM STOC*, pages 734–740, 1994.

[Mau02] Ueli M. Maurer. Indistinguishability of random systems. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 110–132. Springer, 2002.

[Meh79] Kurt Mehlhorn. Some remarks on Boolean sums. *Acta Inf.*, 12:371–375, 1979.

[MMRV02] Konstantin Makarychev, Yury Makarychev, Andrei E. Romashchenko, and Nikolai K. Vereshchagin. A new class of non-Shannon-type inequalities for entropies. *Communications in Information and Systems*, 2(2):147–166, 2002.

[MPR07] Ueli M. Maurer, Krzysztof Pietrzak, and Renato Renner. Indistinguishability amplification. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2007.

[Mut05] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[Nay99a] Ashwin Nayak. *Lower bounds for Quantum Computation and Communication*. PhD thesis, University of California, Berkeley, 1999.

[Nay99b] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *40th IEEE FOCS*, pages 369–377, 1999.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[Nec71] E. I. Nechiporuk. On a Boolean matrix. *Syst.Th.Res.*, 21:236–239, 1971.

[Nis91] Noam Nisan. CREW PRAMs and decision trees. *SIAM J. Comput.*, 20(6):999–1007, 1991. Earlier version in STOC '89.

[NRS99] Noam Nisan, Steven Rudich, and Michael E. Saks. Products and help bits in decision trees. *SIAM J. Comput.*, 28(3):1035–1050, 1999. Earlier version in FOCS '94.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. Earlier version in FOCS '88.

[NW95] Noam Nisan and Avi Wigderson. On the complexity of bilinear forms: dedicated to the memory of Jacques Morgenstern. In *27th ACM STOC*, pages 723–732, 1995.

[Oka00] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *J. Comput. Syst. Sci.*, 60(1):47–108, 2000. Earlier version in STOC '96.

[OP04] Masanori Ohya and Denes Petz. *Quantum Entropy and its Use*. Texts and Monographs in Physics. Springer-Verlag, Heidelberg, 2nd edition, 2004.

[Pau76] Wolfgang J. Paul. Realizing Boolean functions on disjoint sets of variables. *Theor. Comput. Sci.*, 2(3):383–396, 1976.

[Pip80] Nicholas Pippenger. On another Boolean matrix. *Theor. Comput. Sci.*, 11:49–56, 1980.

[Pos11] Alexey Pospelov. Faster polynomial multiplication via discrete Fourier transforms. In *6th CSR*, pages 91–104, 2011.

[PR94] P. Pudlák and V. Rödl. Some combinatorial-algebraic problems from complexity theory. *Discrete Mathematics*, 136(1-3):253 – 279, 1994.

[Pud94] Pavel Pudlák. Communication in bounded depth circuits. *Combinatorica*, 14(2):203–216, 1994.

[PV76] Nicholas Pippenger and Leslie G. Valiant. Shifting graphs and their applications. *J. ACM*, 23(3):423–432, 1976.

[PY82] Nicholas Pippenger and Andrew C.-C. Yao. Rearrangeable networks with limited depth. *SIAM Journal on Algebraic and Discrete Methods*, 3:411–417, 1982.

[Raz85a] A. A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Dokl. Ak. Nauk. SSSR*, 281:798–801, 1985. In Russian. English translation in: Soy. Math. Dokl., 31 (1985), 354-357.

[Raz85b] A. A. Razborov. Lower bounds on monotone network complexity of the logical permanent. *Mat. Zametki*, 37:887–900, 1985. In Russian. English translation in: Math. Notes of the Academy of Sciences of the USSR 37 (1985), 485-493.

[Raz98] Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998. Earlier version in STOC '95.

[Reg11] Oded Regev. Entropy-based bounds on dimension reduction in L_1. arXiv:1108.1283v3, 2011.

[RR12] Ran Raz and Ricky Rosen. A strong parallel repetition theorem for projection games on expanders. In *IEEE Conference on Computational Complexity*, pages 247–257, 2012.

[RRV02] Ran Raz, Omer Reingold, and Salil P. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *J. Comput. Syst. Sci.*, 65(1):97–128, 2002. Earlier version in STOC '99.

[RS03] Ran Raz and Amir Shpilka. Lower bounds for matrix product in bounded depth circuits with arbitrary gates. *SIAM J. Comput.*, 32(2):488–513, 2003. Earlier version in STOC '01.

[RTS00] Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM J. Discrete Math.*, 13(1):2–24, 2000. Earlier version in FOCS '97.

[RW09] Mark D. Reid and Robert C. Williamson. Generalised Pinsker inequalities. In *COLT*, 2009.

[Sav02] Petr Savický. On determinism versus unambiquous nondeterminism for decision trees. *Electronic Colloquium on Computational Complexity (ECCC)*, TR02-009, 2002.

[Sch77] A. Schönhage. Schnelle multiplikation von polynomen über körpern der charakteristic 2. *Acta Informatica*, 7:395–398, 1977.

[Sch81] Arnold Schönhage. Partial and total matrix multiplication. *SIAM J. Comput.*, 10(3):434–455, 1981.

[Sha03] Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1-2):1–22, 2003. Earlier version in CCC '01.

[Sha10] Ronen Shaltiel. Derandomized parallel repetition theorems for free games. In *IEEE Conference on Computational Complexity*, pages 28–37, 2010.

[She11] Alexander A. Sherstov. Strong direct product theorems for quantum communication and query complexity. In *43rd ACM STOC*, pages 41–50, 2011.

[Sto86] Quentin F. Stout. Meshes with multiple buses. In *27th IEEE FOCS*, pages 264–273, 1986.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.

[Str73a] Volker Strassen. Die berechnungskomplexität von elementarsymmetrischen funktionen und von interpolationskoeffizienten. *Numerische Mathematik*, 20:238–251, 1973.

[Str73b] Volker Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 1973:184–202, 1973. In German.

[SV03] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

[SV08] Pranab Sen and Srinivasan Venkatesh. Lower bounds for predecessor searching in the cell probe model. *J. Comput. Syst. Sci.*, 74(3):364–385, 2008. Earlier version in CCC '03.

[Tar89] Gábor Tardos. Query complexity, or why is it difficult to seperate NP $^{\mathrm{a}}$ cap coNP$^{\mathrm{a}}$ from P$^{\mathrm{a}}$ by random oracles a? *Combinatorica*, 9(4):385–392, 1989.

[Uhl74] D. Uhlig. On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements. *Mathematical Notes*, 15:558–562, 1974.

[Ung09] Falk Unger. A probabilistic inequality with applications to threshold direct-product theorems. In *50th IEEE FOCS*, pages 221–229, 2009.

[Val76] Leslie G. Valiant. Graph-theoretic properties in computational complexity. *Journal of Computer and System Sciences*, 13(3):278 – 285, 1976.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *6th MFCS*, pages 162–176, 1977.

[Vio09] Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009.

[vL99] J. H. van Lint. *Introduction to coding theory*, volume 86 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 3rd edition, 1999.

[Š08] Robert Špalek. The multiplicative quantum adversary. In *IEEE Conference on Computational Complexity*, pages 237–248, 2008.

[VW08] Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for polynomials and protocols. *Theory of Computing*, 4(1):137–168, 2008. Earlier version in CCC '07.

[Wat02] John Watrous. Limits on the power of quantum statistical zero-knowledge. In *43rd IEEE FOCS*, pages 459–468, 2002.

[Wat03] John Watrous. PSPACE has constant-round quantum interactive proof systems. *Theor. Comput. Sci.*, 292(3):575–588, 2003. Earlier version in FOCS '99.

[Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM J. Comput.*, 39(1):25–58, 2009. Earlier version in STOC '06.

[Weg91] Ingo Wegener. *The Complexity of Boolean Functions.* Wiley Teubner on Applicable Theory in Computer Science. John Wiley and Sons Ltd., 1991.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *44th ACM STOC*, pages 887–898, 2012.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51st IEEE FOCS*, pages 645–654, 2010.

[Yao77] Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th FOCS*, pages 222–227, 1977.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd ACM FOCS*, pages 80–91, 1982.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *26th IEEE FOCS*, pages 1–10, 1985.

[Yap83] Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983.

[ZY97] Zhen Zhang and Raymond W. Yeung. A non-Shannon-type conditional inequality of information quantities. *IEEE Transactions on Information Theory*, 43(6):1982–1986, 1997.