Kolmogorov Complexity and Formula Size Lower Bounds

Troy Lee

Kolmogorov Complexity and Formula Size Lower Bounds

ILLC Dissertation Series DS-2006-01

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31-20-525 6051
fax: +31-20-525 5206
e-mail: `illc@science.uva.nl`
homepage: `http://www.illc.uva.nl/`

Kolmogorov Complexity and Formula Size Lower Bounds

Academisch Proefschrift

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.mr. P.F. van der Heijden
ten overstaan van een door het college voor
promoties ingestelde commissie, in het openbaar
te verdedigen in de Aula der Universiteit
op woensdag 11 januari 2006, te 12.00 uur

door

Troy Jeffrey Lee

geboren te Houston, Verenigde Staten van Amerika.

Promotiecommissie:

Promotor:	Prof.dr. H.M. Buhrman
Overige leden:	Prof.dr. L. Fortnow Dr.ir D. van Melkebeek Dr. L. Torenvliet Prof.dr.ir P.M.B. Vitányi Prof.dr. N.K. Vereshchagin

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

ISBN: 90–6196–534–9

“I wish I was a headlight on a north-bound train;
I’d shine my light through the cool Colorado rain.”

I Know You Rider
Traditional Song

Contents

Acknowledgments

First and foremost, I would like to thank Harry Buhrman. Harry took me out of his quantum computing course and under his wing to supervise my masters thesis and then offered me a PhD position. In the time since, he has continued to show his support and confidence in me throughout the trials and tribulations that come with banging your head against a wall until it turns into a thesis. He has always enthusiastically shared his latest ideas, and provided both direction and distraction. And all I had to do was keep letting him win at ping-pong...

This thesis owes a lot to my coauthors: Harry Buhrman, Lance Fortnow, Sophie Laplante, Dieter van Melkebeek, Andrei Romashchenko, Mario Szegedy, and Kolya Vereshchagin. It is papers with them which make up this thesis; but more than that, I want to thank them for all the good times that went along with creating these papers. It has been wonderful to see the power of interaction firsthand in our collaboration.

I also want to thank the members of my committee: Lance Fortnow, Dieter van Melkebeek, Leen Torenvliet, Paul Vitányi, and Kolya Vereshchagin, for reading and providing many helpful comments on this thesis.

A change of air is always good to inspire new ideas. I want to thank Peter Bro Miltersen for inviting me to the crisp fall air of Aarhus where parts of Chapter 5 were born; Mario Szegedy for a delightful, delectable, de-lemmaful week at Rutgers; and Sophie Laplante—whose hospitality cannot be upper bounded—for always welcoming me in Paris.

Back in Amsterdam, I couldn't hope for better companions than Robert Špalek and Falk Unger, and I was lucky to have a perfect proofreader down the hall in Ronald de Wolf. More generally, Ronald could be consulted for a sound opinion on everything from French farm subsidies to the best red wine under five euros at the Albert Heijn.

I had wonderful office mates in Peter Grünwald and Michal Koucký. I would like to thank all the current and past hallway inhabitants for the interesting discussions—scientific and otherwise—we've had over the years: Rudi Cilibrasi, Mart de Graaf, Hartmut Klauck, Steven de Rooij, John Tromp, Paul Vitányi, and Stephanie Wehner. Special thanks to John and Steven for help in writing the samenvatting. Having an office strategically situated next to our printer—which "loves jammin'" like Bob Marley—had the advantage of bringing frequent visits by Matthijs Mourits and his lively wit.

I want to thank the greater complexity community for being such a friendly and welcoming group of people. Among others, I have had many enjoyable conversations with: Scott Aaronson, Eric Allender, Andris Ambainis, Luis Antunes, Andrej Bogdanov, Andrew Childs, Matthias Christandl, Christoff Dürr, Steve Fenner, Anna Gal, Bill Gasarch, Danny Gutfreund, Kristoffer Hansen, John Hitchcock, Neeraj Kayal, Julia Kempe, Eike Kiltz, Boaz Leskes, María López-Valdés, Jack Lutz, Frédéric Magniez, Elvira Mayordomo, Ilan Newman, Ryan O'Donnell, Oded Regev, Nitin Saxena, Jacobo Torán, Luca Trevisan, Emanuele Viola, Hoeteck Wee, and Enav Weinreb.

Thanks to the Boavida family who gave me a place to go home to on this continent. À minha querida Maria, que me deu novas línguas—ainda não sei como dizer obrigado pelos últimos cinco anos. Tenho que aprender mais...

Most of all, I want to thank my parents. Although they claim not to understand anything past page four, they have done more for my love of science than anyone else. I cannot thank them enough for all the support and encouragement they have given me over the years.

Troy Lee
Amsterdam, December 2005

Chapter 1

Preface

This thesis consists of two parts, the first on Kolmogorov complexity, and the second on formula size lower bounds. Kolmogorov complexity is an elegant way to mathematically define the concept of randomness, taking the definition that a random string is one which has no shorter description than the length of the string itself. Formulas are circuits composed out of AND, OR, and NOT gates, where each gate has exactly one outgoing wire. They are interesting to study because they are a reasonably powerful model of computation for which we are still able to prove nontrivial lower bounds.

These two parts are motivated by two of the grand open problems in theoretical computer science. The first of these is the power of randomness, and the question of whether or not we can always efficiently simulate algorithms which base their decisions on the flip of a coin by more impoverished algorithms which have no coin to flip. The second problem is the question of whether or not P = NP, where P is the set of problems which can be efficiently solved, and NP is the set of problems whose solution can be efficiently verified given the solution. Most researchers currently believe that P $\neq$ NP, and to do this one seemingly needs to show a problem in NP which requires superpolynomial time to solve. Showing lower bounds on a weaker model like formula size can be seen as a modest step towards this goal.

Interestingly enough, these two problems—that of the power of randomness and that of proving lower bounds—turn out to be intricately related by the beautiful theory of hardness versus randomness tradeoffs, which we will encounter in Chapter 4. This theory says that to show randomized algorithms can be simulated deterministically, it suffices to show *lower bounds* in a model closely related formulas, that of circuits [NW94]. More recent work shows that in fact proving lower bounds is necessary in order to simulate randomized algorithms deterministically [KI03].

We begin the thesis with the words of Herman Melville: “But no more of this blubbering now, we are going a-whaling, and there is plenty of that yet to come.”

Part I: Kolmogorov Complexity

Chapter 2

Kolmogorov Complexity

When is a sequence of events random? In daily life, the word random is often invoked to describe tomorrow's weather, the flip of a coin, or the shuffle function of a music player. Despite this familiarity, or perhaps because of it, mathematicians struggled for centuries to come up with a satisfactory definition of randomness.

The earliest attempts at defining randomness were directly inspired by intuitions from gambling. Let us think of a device commonly thought of as having random outcomes, say a roulette wheel. What justification do we have for thinking the spin of the roulette wheel is a random event? One reason is that, after centuries of trying, no one has yet found a gambling strategy for predicting where the ball will land and consistently winning money.

Following this intuition, let us take the preliminary definition that a sequence of events is random if there is no successful gambling strategy for predicting its outcomes. This "Law of Excluded Gambling Strategy" approach to randomness was taken by von Mises in 1919 [vM19]. Let us explore for a moment what implications such a definition has. It is generally assumed that the outcomes of a coin flip are random with equal probability of being heads or tails—for interesting discussions on the validity of this assumption in practice see [Kel86] and [DHM04]. Under this assumption, a bookmaker would place the odds on heads at 1:1. If you bet one forint, the currency of combinatorics [Lov93], on heads and are correct then you win one forint, otherwise you lose it.

Let us first examine the fate of a very simple strategy—always bet heads. This strategy should not make us money. That means that the probability of heads can be at most 1/2. Symmetrically, as the strategy "always bet tails" does not make us money either, we find the probability of heads is the same as that of tails and equal to 1/2.

Let us now look at a slightly more complicated strategy—on odd numbered flips we bet heads and on even numbered flips we bet tails. Again this strategy should not make us money. This tells us that the probability of consecutive coin flips coming up heads–tails is at most 1/4. Similarly, we find that all four possible outcomes for two flips of a coin have probability equal to 1/4.

We can already see that this simple definition has implications which agree with our intuitions. More properties of a sequence of coin flips can be derived by considering more sophisticated strategies. We now arrive at a crucial point: Exactly what gambling strategies do we allow?

At one extreme, we could allow all possible strategies, that is any function from the natural numbers into the set of coin flip outcomes $\{0, 1\}$, with 0 representing tails and 1 heads. If we allowed all functions, however, then no sequence of coin flips would be random, as there is always a function which exactly predicts its outcomes!

At the other extreme, we could only consider strategies of the simple form given as examples earlier. That is, we could take some finite string $x \in \{0, 1\}^n$ of length n, and in the i^{th} coin flip we predict the outcome to be letter in position $i \mod n$ of x. Strategies of this form will not be able to win any money betting against the number $0.(0)(1)(10)(11)(100)(101)(110)(111)\ldots$, known as Champernowne's binary constant. This number is formed by consecutively concatenating all natural numbers written in binary. The reason that no such strategy would win money against Champernowne's binary constant is that the frequency any k-bit sequence occurs in Champernowne's binary constant is 2^{-k}. Numbers satisfying this property are known as normal. We certainly would not consider Champernowne's constant to be random, however, as there is still a very simple gambling strategy which would make our fortune.

A very different approach to randomness was taken, independently and within a few years of each other, by Solomonoff, Kolmogorov, and Chaitin. See the history and references section of Chapter 1 of [LV97] for more background. The intuition behind their approach is that a random sequence is one which has no easily discernible patterns, no structure which we could take advantage of to shortly describe the string. A random sequence, then, is defined as one which has no shorter description than the the length of the sequence itself. To formalize this definition, we are again faced with a question similar to that of what gambling strategies do we allow, namely, what types of descriptions do we allow?

Both of these approaches seem to call for some notion of computation. Indeed it was the theory of computation developed by Turing and others in the 1930's that both of these approaches adopted for their formalization. We should mention that the "Law of Excluded Gambling Strategy Approach", when excluding all Turing computable strategies, still does not lead to a satisfactory definition of randomness. The approach based on descriptions, on the other hand, when taking the set of descriptions to be programs on a Turing machine, does lead to a definition of randomness which agrees with our intuitions.

Turing's theory of computation is remarkably robust and some 70 years after its formulation has resisted all challenges to remain the standard definition of what is in principle computable. As computers became a reality, however, interest shifted from what could be computed in principle, to what could be computed *efficiently.* Thus the field of computational complexity was born. With it were born a multitude of complexity classes capturing the hardness of computational problems around us. Salient among these complexity classes are two which will play a major role in this thesis: NP the class of problems which have solutions which can be efficiently verified, and BPP, the class of problems whose solutions can be found with high probability with the help of flipping a coin.

These new notions of computation freshly raised questions about the definition of randomness. After all, what looks random to a computationally restricted observer will be different than for a Turing machine. To illustrate this point, let us once again enter the casino in a thought experiment inspired by Oded Goldreich [Gol01]. The roulette wheel spins in front of us, a whirl of red and black labelled with the numbers 0 to 36. Unfortunately, we don't have much time to make our fortune, and thus we are betting only on the numbers 0–36 where the big payoff lies. The probability that the wheel lands on any number we choose is $1/37$, but the casino gives itself some advantage in setting the odds at 35:1. We are allowed to place our bets even after the ball has been set in motion— it is still random to us, right?

Imagine we enter the casino covertly carrying a stopwatch. We clock the period of the ball as it makes its turn, and, knowing the radius of the wheel, determine how fast the ball is moving. Would the casino still bet that we cannot predict the outcome with probability better than $1/36$? Say we now also enter the casino with a tiny computer which we feed the speed of the ball, and also the speed of the rotating wheel, and the computer then, knowing approximately the coefficient of friction on a regulation roulette wheel from the one we practiced with at home, computes the deceleration of the ball and at what point it will fall into the inner part of the wheel. Will the casino make the same bet against us with all this information? Can this hi-tech observer now predict the outcome with probability better than chance?

Casinos would certainly not be willing to make the same bet to such an observer, illustrated by the fact that no such electronic devices are allowed in the casino, and anyone caught using one will be promptly thrown out, or worse if we believe the movies. Predicting the outcome of a roulette wheel in such a fashion has been put successfully into practice at least once, by a group of physics graduate students. See [Bas00] for their entertaining story.

As the rolls of the wheel might be predictable by a more sophisticated player but not by a normal one, similarly a sequence which is not random with respect to a Turing machine might still appear random to a less sophisticated polynomial time algorithm. This is the basic idea behind the theory of "pseudorandomness".

The main object of study in pseudorandomness are probability distributions. Like the "Law of Excluded Gambling Strategy" approach to randomness, pseudorandomness is defined in terms of a predictor. Namely, a distribution X on n-bit strings is called pseudorandom if there is no *efficient* algorithm A and which can, on seeing the first i bits of a string drawn from X, predict the $i+1$ bit with advantage, on average. More precisely, X is pseudorandom if for every efficient algorithm A

$$|\Pr[A(x_1x_2\dots x_i) = x_{i+1}] - 1/2| \leq \epsilon$$

where x is chosen according to the distribution X and i is chosen uniformly between 0 and $n-1$. Here ϵ should be thought of as a negligibly small number. To return to the casino, this means that if the truly random roulette wheel were replaced instead with the pseudorandom source X, we would still not be able to predict the outcomes of the wheel with probability much better than chance.

In the first part of this thesis, we will be looking a different approach to pseudorandomness obtained by scaling down the theory of Kolmogorov complexity into the efficient domain. We will often refer to this as resource bounded Kolmogorov complexity. This theory is interesting in its own right, and is related to the prediction approach to pseudorandomness via the rich theory of "hardness vs. randomness" tradeoffs—this tradeoff gives a procedure to convert a string which is random in the sense of Kolmogorov complexity, that is a "hard" string, into a distribution which is pseudorandom. Not surprisingly, "hardness vs. randomness" tradeoffs will play a key role in our results. Before we get into the details of resource bounded Kolmogorov complexity, however, we first review the classical theory.

2.1 Turing Machines

In trying to model how an idealized mathematician works, Turing came up with an idea that forms the foundation of modern day complexity theory—the Turing machine. Turing imagined a mathematician with an infinite one dimensional piece of paper, divided into squares like a child's arithmetic book. In each step, the mathematician writes one of a finite number of symbols in a square of the paper, reads a symbol written on a square of the paper, moves to an adjacent square, or does nothing.

Thus a Turing machine is a computing device which has a number k of one-way infinite one-dimensional pieces of paper, known as tapes, divided into squares called cells. The first of these tapes is known as the *input tape*, where the problem is written, and the last of the tapes is known as the *output tape*, where the result is written. The other $k-2$ tapes can be thought of as the mathematician's scratch paper. The cells of the tapes can be blank or contain a symbol from some finite alphabet Σ. We will always assume the alphabet is just $\{0,1\}$. On each tape the Turing machine has what is called a head. This can be thought of as the tip of the mathematician's pencil. At any one time, a head sits on a particular cell and can read the symbol which is written on that cell, write a symbol onto that cell, move to the left or right or stay put. The behavior of the Turing machine is governed by a fixed finite program which directs the actions of the heads in response to what is written on the cells the head is reading.

Initially, a finite string is given on the input tape, and the worktape and output tape are blank with their heads sitting on the leftmost cells. The string initially written on the input tape will be called the input. The output of the machine is what is written on the output tape when the machine enters a special state called the final or halting state.

Given an input, the behavior of a Turing machine is completely described by its program. Thus we will identify a Turing machine with its program. A program is simply a binary string, thus we can further identify programs with binary strings and therefore natural numbers. Notice that a valid program may have a particular form, thus not all natural numbers will encode valid programs. As we will often make reference to the correspondence of binary strings and natural numbers, we now fix the bijection we use which associates a string with its index in a lexicographical ordering of strings.

$$(\varepsilon, 0), (0, 1), (00, 2), (01, 3), (10, 4), (11, 5), \dots. \tag{2.1}$$

Here ε denotes the empty string. Notice in this way, the number n is encoded by a string of length $\lfloor \log n + 1 \rfloor$. We take this opportunity to say that the length of a string x will be written $\ell(x)$.

A simple observation with deep consequences is that there is a Turing machine which given input a natural number n can decode this number into a program and decide if it is formatted as a valid program or not. This leads to what is called an effective enumeration of Turing machines. We can enumerate Turing machines as $T_1, T_2, T_3, \dots$ where the program of machine T_i is the i^{th} valid program in lexicographical order. As we shall now see it is this effective enumeration of Turing machines which forms the foundation of Kolmogorov complexity.

2.2 Kolmogorov complexity

We have informally stated the Kolmogorov complexity of a string x as the length of a shortest description of x. As in the case of gambling strategies, to formalize this definition we need to specify exactly what we mean by a "description". Most generally, a description can be thought of as a partial function $f : \mathbb{N} \to \mathbb{N}$. We would then define the Kolmogorov complexity with respect to f to be

$$C_f(x) = \min_p \left(\{ l(p) : f(p) = x \} \cup \{\infty\} \right).$$

It is also interesting to consider a conditional version of Kolmogorov complexity, where we are given some "advice" string y to use for free in our description of x. Before we define conditional complexity, we first define the very useful notion of a pairing function:

2.2.1. Definition (Pairing function). *We define a pairing function $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. We want such a pairing function to be easily computable and* *easily invertible. Given $x, y \in \{0,1\}^*$ we define $\langle x, y\rangle = 0^{l(x)}1xy$. To decode this function, we first count the number of zeros before the first one. We then know the length of x, and can therefore separate the subsequent concatenation of x and y.*

Conditional complexity is now defined as before, with the function also given the advice string in the input.

$$C_f(x \,|\, y) = \min_p \left(\{l(p) : f(\langle p, y\rangle) = x\} \cup \{\infty\}\right).$$

Notice that $C_f(x)$ is simply $C_f(x \,|\, \varepsilon)$, where ε is the empty string.

To have a meaningful theory we would like to define the complexity of a string with respect to a class of description methods, such as the set of all C programs, rather than just a single function f. There is a standard method to construct a Kolmogorov measure for a class of description methods. We describe this procedure in a rather general fashion so that we can later directly apply it the various forms of resource-bounded Kolmogorov complexity we define.

Let $\mathcal{F} \subseteq \mathbb{N}^{\mathbb{N}}$ be a countable set of partial functions $f : \mathbb{N} \to \mathbb{N}$. The way we move from talking about complexity with respect to a single function to talking about the complexity with respect to a class of functions is to construct a *Kolmogorov minimal element* for the class.

2.2.2. Definition (Kolmogorov minimal element). *We say that $\mathcal{F}$ has a Kolmogorov minimal element if there is some f such that*

- $f \in \mathcal{F}$
- *for all $f' \in \mathcal{F}$ there is a constant c such that for all $x, y \in \mathbb{N}$*

$$C_f(x \,|\, y) \leq C_{f'}(x \,|\, y) + c.$$

The existence of a Kolmogorov minimal element for $\mathcal{F}$ makes it asymptotically meaningful to talk about Kolmogorov complexity relative to the class $\mathcal{F}$. Namely, we fix one Kolmogorov minimal element f for the class $\mathcal{F}$, and define the Kolmogorov complexity relative to the class $\mathcal{F}$ to be the complexity with respect to f. The choice of which particular Kolmogorov minimal element we take will affect our results by at most a constant. This is what [LV97] call the invariance theorem.

We now make this discussion more concrete by instantiating it for the set of partial recursive functions.

2.2.3. Theorem. *There is a Kolmogorov minimal element for the class of partial recursive functions.*

Proof: The crucial property we need is that there is a recursive enumeration $\phi_1, \phi_2, \phi_3, \ldots$ of partial recursive functions. With this enumeration in hand, we can construct a Kolmogorov minimal element ϕ as follows: ϕ interprets its input as a triple $\langle i, p, y \rangle$, generates ϕ_i via our recursive enumeration, and simulates the running of $\phi_i(\langle p, y \rangle)$. Thus the output of $\phi(\langle i, p, y \rangle)$ is $\phi_i(\langle p, y \rangle)$.

Let ψ be a partial recursive function. Then ψ appears somewhere in our enumeration of partial recursive functions, say $\psi = \phi_i$. If $\phi_i(\langle p, y \rangle) = x$ then $\phi(\langle i, p, y \rangle) = x$. Thus there is a constant c, which we can take to be $c = 2\lfloor \log i + 1 \rfloor + 1$ such that

$$C_\phi(x \,|\, y) \leq C_\psi(x \,|\, y) + c$$

for all x, y. □

We now fix such a Kolmogorov minimal element ϕ, and define the Kolmogorov complexity to be $C(x \,|\, y) = C_\phi(x \,|\, y)$. Which particular minimal element we take only matters up to an additive constant in our definition.

In other words, up to an additive constant, it does not matter if we define our complexity with respect to C programs or Java programs. Either way, we can write for example a compiler for Java programs in C and therefore use any Java program at only a fixed constant size penalty in description size, namely the length of the compiler.

2.2.1 Prefix-free complexity

Thus far, our discussion has focused on randomness and the problem of how to define when a binary sequence is random. What we have arrived at, however, gives us a way of telling not only when a sequence is random, but also a way of quantifying the amount of information in an individual string. Such a theory has further reaching implications than our initial purpose.

One such application is the potential to formalize intuitive statements about simplicity and complexity. One such statement is Occam's razor: "if presented with a choice between indifferent alternatives, then one ought to select the simplest one". Occam's razor slices through a thicket of theories to select the simplest one compatible with observations. If a theory is simple in any recognizable way it will have low Kolmogorov complexity, thus we now have a way to make this statement precise.

According to Occam's razor, if we are completely ignorant, that is all theories appear indifferent to us, then we would like to bias simple theories by giving them a higher probability of being selected. A natural way to do this would be to give theory x initial probability $2^{-C(x)}$. A problem with this is that $2^{-C(x)}$ is not actually a probability distribution as $\sum_x 2^{-C(x)}$ diverges. Indeed, consider only strings of the form $x_n = 0^n$. Each of these strings can be described by $\log n + c$ many bits for some constant c. Thus $\sum_{x_n} 2^{-C(x_n)} \geq c' \sum_n 1/n$ which diverges.

This problem can be remedied by considering prefix-free codes. A set $\mathcal{C} \subseteq \{0,1\}^*$ is called *prefix-free* if for all $x, y \in \mathcal{C}$ there is no nonempty string z such that $x = yz$. Imagine a binary tree whose nodes are labelled from $\{0,1\}^*$ according to the path taken from the root to that node with "left" denoted 0 and "right" denoted 1. If a set $\mathcal{C}$ is prefix-free this means that if an element of $\mathcal{C}$ labels a node of the tree, then no element of the set $\mathcal{C}$ can label a descendant of this node.

An important property of prefix-free codes is what is known as the Kraft Inequality.

2.2.4. Theorem (Kraft). *Let $\mathcal{C} \subseteq \{0,1\}^*$ be a prefix-free set. Then*

$$\sum_{x \in \mathcal{C}} 2^{-l(x)} \leq 1.$$

Proof: We can identify a binary string x with a real number ω_x in the interval $[0,1]$ where $\omega_x = \sum_i x_i 2^{-i}$. Associate to each $x \in \mathcal{C}$ the interval $I_x = [\omega_x, \omega_x + 2^{-l(x)})$. Then the interval I_x is of length $2^{-l(x)}$, and by the prefix-free property of $\mathcal{C}$ the intervals I_x, I_y are disjoint for distinct $x, y \in \mathcal{C}$. The result now follows as all the intervals lie in $[0,1]$. □

To define prefix-free complexity, we restrict ourselves to work with partial recursive functions which are prefix-free.

2.2.5. Definition (prefix-free partial recursive function). *A partial recursive function ϕ is called prefix free if whenever $\phi(p) < \infty$ and $\phi(q) < \infty$ then p is not a proper prefix of q.*

Although it is undecidable to determine if a given partial recursive function ϕ is prefix-free, we are able to construct a recursive enumeration which includes all prefix-free partial recursive functions, and such that every function in the enumeration is prefix free. One way to do this is to use our enumeration of partial recursive functions $\phi_1, \phi_2, \phi_3, \ldots$, but modify each ϕ into a function ψ such that ψ is prefix-free and if ϕ was already prefix free, then $\psi = \phi$.

Before we describe our enumeration of prefix-free function, we first define a very useful concept called dovetailing.

2.2.6. Definition (Dovetail). *Say that we are given a Turing machine T, and we wish to find a halting program of T, if one exists.* Dovetailing *is a procedure to do this. In the first stage we run the empty program on T for one time step. In stage k we run all programs of length i for j time steps where $i + j = k$, with i, j nonnegative. If T has a halting program this procedure will eventually find it.*

Given a partial recursive function ϕ, we dovetail the running of all programs on ϕ. Initially we set ψ to be undefined for all inputs. We then construct ψ inductively.

- Stage 1: Dovetail the running of ϕ on all programs. Say that p is the first program which halts in this process. Set $\psi(p) = \phi(p)$. This is the end of stage 1. If p is the empty program, our construction of ψ is done. Otherwise, we go on to stage 2.

- Stage k: Say that in previous stages we have seen the programs $p_1, \ldots, p_k$ halt in our dovetailing of ϕ. We continue dovetailing ϕ until a we see new program p_{k+1} halt. If p_{k+1} is a prefix of some p_i for $1 \leq i \leq k$ then we leave $\psi(p_{k+1})$ undefined, and our construction of ψ is done. Otherwise, we set $\psi(p_{k+1}) = \phi(p_{k+1})$ and go on to stage $k+1$.

It is clear from this construction that every ψ is prefix-free and if ϕ is already prefix free then $\psi = \phi$.

2.2.7. THEOREM. *There is a prefix-free partial recursive function ψ such that for any prefix-free partial recursive function ψ' there is a constant c such that for all x, y*

$$C_\psi(x \mid y) \leq C_{\psi'}(x \mid y) + c.$$

Proof: Now that we have an enumeration $\psi_1, \psi_2, \psi_3, \ldots$ of prefix-free partial recursive functions, we can define a minimal element for the prefix-free partial recursive functions in the same way as before. Define

$$\psi(\langle i, p, y \rangle) = \psi_i(\langle p, y \rangle).$$

As the function ψ' is a prefix-free partial recursive function it appears somewhere in our enumeration, $\psi' = \psi_i$ for some i. As before, we can take the constant c to be $c = 2\lfloor \log i + 1 \rfloor + 1$. □

As before we now take a Kolmogorov minimal element for the prefix-free partial recursive functions ψ and define $K(x \mid y) = C_\psi(x \mid y)$. We use the notation K to distinguish from the plain Kolmogorov complexity denoted with C.

We now discuss an alternative way of defining prefix-free complexity in terms of self-delimiting Turing machines. In some sense this approach is more intuitive as we can physically enforce the condition that the set of halting programs is prefix-free by modifying the behavior of Turing machines and the definition of halting programs.

The head on the input tape of a self-delimiting Turing machine is restricted to only move from left to right and not the other way around. We change the definition of a halting program to be only those programs where the machine halts while reading the last marked cell of the input, the cell before the blank. Alternatively, we can imagine that the machine is always given an infinite string as input and define a halting program to be the finite prefix which the machine has read when it halts. It is clear that both of these modifications enforce the condition that if p is a halting program and q is a halting program, then p is not a proper prefix of q.

We can enumerate self-delimiting Turing machines just as in our enumeration of regular Turing machines, and so define a Kolmogorov minimal element in the standard fashion. We now see that this definition in terms of self-delimiting Turing machines is equivalent to the definition in terms of prefix-free partial recursive functions. For this argument, let $K_1(x \mid y)$ be the complexity of x given y with respect to the definition in terms of prefix-free partial recursive functions, and let $K_2(x \mid y)$ be the complexity with respect to the definition in terms of self-delimiting Turing machines. As each self-delimiting Turing machine computes a prefix-free function, it appears somewhere in our enumeration $\psi_1, \psi_2, \psi_3, \ldots$ of prefix-free functions, thus it follows from the existence of a Kolmogorov minimal element for prefix-free functions that there is a constant c such that $K_1(x \mid y) \leq K_2(x, y) + c$.

Now for the other direction, we show how to simulate the behavior of a prefix free function on a self-delimiting Turing machine. Let ϕ be a prefix-free function, and let T be a Turing machine which computes ϕ. Say that we wish to evaluate $T(p)$ on a self-delimiting machine M.

- Step 1: Before M reads any bit of the input it begins running all programs on T in a dovetail fashion. If T first halts on the empty program, then M halts with the output of $T(\varepsilon)$ without having read any of the input, and the simulation is finished. Otherwise, if T halts on some other program q not equal to the empty program, then M reads the first bit of the input p_1 of p and moves on to step 2.

- Step k: Suppose that M has now read $p_1 \ldots p_{k-1}$ the first $k-1$ bits of p. M now runs in dovetail fashion all programs on T with prefix $p_1 \ldots p_{k-1}$. If $p_1 \ldots p_{k-1}$ is the first program to halt, then M outputs $T(p_1 \ldots p_{k-1})$ and the simulation terminates. Otherwise, if T first halts on some program $p_1 \ldots p_{k-1}q$ with q not the empty string, then M reads the next bit p_k of p and moves to step $k+1$.

Thus, in particular, we can simulate the behavior a Kolmogorov minimal prefix-free function by a self-delimiting Turing machine and so $K_2(x \mid y) \leq K_1(x \mid y) + c$.

Now that we have laid out the foundations of Kolmogorov complexity, we are ready to begin building up the theory. Central to the construction of the theory are four fundamental theorems with widespread application, both within Kolmogorov complexity and more generally in theoretical computer science. We refer to these four fundamental theorems as the four pillars.

2.3 Four pillars of Kolmogorov complexity

We identify four “pillars” of Kolmogorov complexity. These are fundamental theorems of Kolmogorov complexity which are widely applied to other areas of computer science and to more advanced theorems of Kolmogorov complexity. A main goal of this thesis will be to see what, if any, analogues of these theorems hold as we move to the resource bounded setting and to the study of Kolmogorov complexity with error.

2.3.1 First pillar: existence of incompressible strings

2.3.1. THEOREM. *For every length n and any string $z \in \{0,1\}^*$ there exists $x \in \{0,1\}^n$ such that $C(x \mid z) \geq n$.*

Proof: The proof is by simple counting. There are 2^n strings of length n and only $2^n - 1$ many potential programs of length less than n. Thus at least one string $x \in \{0,1\}^n$ must require a program of length at least n, even given z. □

This theorem is surprisingly useful for its simplicity. It is the basic principle behind what is known as the incompressibility method. The incompressibility method shows the existence of an object with a certain property by considering a Kolmogorov random object which is incompressible. If this random object did not possess the desired property the argument then shows that the object could be given a shorter description, thus arriving at a contradiction.

As a demonstration of the first pillar in action, we give a simple example of the incompressibility method to what is known as Ramsey theory. Loosely speaking, the thesis of Ramsey theory is that any large object necessarily possesses patterns. Recall that in a graph a clique is a set of vertices which are all mutually connected by edges, and an independent set is a set of vertices with no edges between them. The hallmark theorem of Ramsey theory, shown in 1929 by Frank Ramsey [Ram29], is that for any k there is some finite n such that all graphs with at least n vertices contains either a clique of size k or an independent set of size k. For example, in any group of six people at least three people all know each other or all do not know each other. The following theorem, originally proved in 1947 by Erdős using the probabilistic method, lower bounds how large a graph must be before we are sure it contains either a clique or independent set of a certain size.

2.3.2. THEOREM (ERDŐS). *For every n, there exists a graph G on n vertices such that G has neither a clique nor an independent set of size larger than $2\lfloor \log n + 1 \rfloor + 1$.*

Proof: We identify graphs on n vertices with strings of length $\binom{n}{2}$ in the natural way, letting the string represent the characteristic sequence of the $\binom{n}{2}$ possible edges in the graph. Let $k = 2\lfloor \log n + 1 \rfloor + 1$, and let p be a fixed program which describes the procedure given below for reconstructing a graph. Let G be a graph on n vertices satisfying $C(G \mid k, n, p) \geq \binom{n}{2}$ which we know exists by the first pillar. We show that if G has a clique or independent set of size k then we can compress G, a contradiction to its definition.

Suppose that G has a clique or independent set of size k. We use one bit of description to say which one is the case, and then describe G as follows. We give a listing of the k vertices of the clique or independent set. Using the bijection given in Equation (2.1), this costs $k\lfloor \log n + 1 \rfloor$ bits. We further explicitly give the adjacency listing for the remaining $n(n-1)/2 - k(k-1)/2$ edges. Thus our total description is of length

$$\frac{n(n-1) - k(k-1)}{2} + k\lfloor \log n + 1 \rfloor + 1.$$

This length cannot be smaller than $n(n-1)/2$ by definition of G, thus we find

$$k \leq 2\lfloor \log n + 1 \rfloor + 1.$$

□

The incompressibility method is a nonconstructive proof technique—although we have shown the existence of a graph with no large clique or independent set, and this proof can be easily modified to show that *most* graphs have no large clique or independent set, we have not provided an efficient algorithm for constructing such a graph. Finding explicit constructions of graphs whose existence is given by Theorem 2.3.2 remains a major open problem of Ramsey theory.

2.3.2 Second pillar: language compression theorem

2.3.3. Theorem. *For every recursively enumerable set A, $C(x) \leq \log |A^{=n}| + O(\log n)$ for all $x \in A$ of length n.*

Proof: Fix n. Let M be a machine which enumerates A. We dovetail the running of M over all strings $x \in \{0,1\}^n$. For each string $x \in A$ the machine M will eventually halt and say ‘accept’. Furthermore, these computations will halt in a definite order. We therefore can describe each $x \in A^{=n}$ by an index $i \in [|A^{=n}|]$ saying that x is the ith string of length n that M will accept when run in dovetail fashion on strings of length n. The description requires $\log |A^{=n}|$ bits to specify the index, and $O(\log n)$ bits to describe n and the machine M. □

We now give an example of an application of the language compression theorem.

2.3.4. Theorem (Adleman [Adl78]). BPP $\subseteq P$/poly

Proof: Let L be a language in BPP. Then there is a polynomial time machine M such that if $x \in L$ then $\Pr_r[M(x,r) = 1] \geq 2/3$ and if $x \notin L$ then $\Pr_r[M(x,r) = 1] \leq 1/3$, where the probability is taken over strings $r \in \{0,1\}^{p(n)}$ for a polynomial p. We can amplify the success probability by on input x outputting the majority vote of n^2 independent runs of M. Call the machine which does this M'. We then have by a Chernoff bound that for large enough n, if $x \in L$ then $\Pr_r[M'(x,r) = 1] \geq 1 - 2^{-2n}$ and if $x \notin L$ then $\Pr_r[M(x,r) = 1] \leq 2^{-2n}$ where now the probability is taken over strings r of length $n^2p(n)$. Take $r \in \{0,1\}^{n^2p(n)}$ satisfying $C(r) \geq n^2p(n)$. We claim that for all $x \in \{0,1\}^n$ the string r gives the "right" answer, that is, that $M(x,r) = 1$ if and only if $x \in L$.

Suppose for contradiction that r gives the wrong answer for some x. For concreteness, say that $x \in L$ and $M(x,r) = 0$. This means that r belongs to a "small" set, a set of size $2^{n^2p(n)-2n}$, the set of random strings which give the wrong answer. Furthermore, given x membership in this set can be checked in polynomial time—we just run $M(x,r)$ for polynomially many steps and see if $M(x,r) = 0$. Thus by the language compression theorem, $C(r \,|\, x) \leq n^2p(n) - 2n + O(\log n)$. As we can explicitly give x with n bits, this means $C(r) \leq n^2p(n) - n + O(\log n)$, a contradiction. □

2.3.3 Third pillar: source compression theorem

The previous theorem addressed the compression of languages. We now look at the compression of probability distributions. The goal now becomes to give elements with large probability short descriptions, in particular we wish to give an element with probability p a description of length about $-\log p$. The next theorem states that this is indeed possible for enumerable distributions. To explain what we mean by enumerable, we first develop some terminology.

2.3.5. Definition. A real function $f : \mathbb{N} \to \mathbb{R}$ is said to be enumerable (from below) if there is a recursive function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{Q}$ such that $\lim_{k \to \infty} g(x,k) = f(x)$, and g is nondecreasing in k. A real function $f : \mathbb{N} \to \mathbb{R}$ is said to be recursive if there is a recursive function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{Q}$ such that $|f(x) - g(x,k)| < 1/k$.

Now we have defined what it means for a function with domain $\{0,1\}^*$ to be enumerable, but what we are ultimately interested in are probability distributions which have domain *subsets* of $\{0,1\}^*$. The notion of *cylinders* gives a simple way to extend the definition of enumerability to probability distributions. For a finite string $x \in \{0,1\}^*$ the cylinder $\Gamma_x \subseteq \{0,1\}^*$ is defined as

$$\Gamma_x = \{\omega \in \{0,1\}^* : \omega_{1:l(x)} = x\}$$

where $\omega_{1:l(x)}$ denotes the first $l(x)$ many bits of ω. We identify a function μ defined on cylinders, such as a probability distribution, with a function μ' defined on $\{0,1\}^*$ by $\mu(\Gamma_x) = \mu'(x)$. If μ' is enumerable then we say μ is enumerable. We will now systematically use this shorthand and drop the subscript.

2.3.6. Definition. A function $\mu : \{0,1\}^* \to \mathbb{R}$ is a semi-probability distribution if and only if

- $\mu(\varepsilon) \leq 1$
- $\mu(x) \geq \mu(x0) + \mu(x1)$.

If μ satisfies the first item with equality then we call μ a probability distribution.

2.3.7. Theorem (Levin). *Let μ be an enumerable semi-probability distribution. Then there is a constant c such that for all x,*

$$K(x) \leq -\log \mu(x) + c.$$

Proof: For clarity we will first describe the encoding assuming that μ is recursive. We will then show how to modify this construction for μ enumerable.

The basic idea is to associate with every element of the support of μ an interval of $[0,1)$ such that the intervals are disjoint and the interval corresponding to x has length $\mu(x)$. The encoding of x will be the left endpoint of the interval associated with x given to $-\log \mu(x)$ many bits of precision.

More formally let the interval of $[0,1)$ corresponding to x be

$$I_x = \left[\sum_{y<x} \mu(y), \sum_{y \leq x} \mu(y)\right).$$

We can do this as μ is a semi-probability distribution and so $\sum_y \mu(y) \leq 1$. We would like to describe x by the left endpoint of I_x and as our goal is to give x a description of length about $-\log \mu(x)$ we would like to specify the left endpoint to only this many bits of precision. We now run into a problem as in general truncating the endpoint to this precision could cause our intervals to overlap, and therefore we would lose the desired prefix-free quality of the encodings.

A solution to this problem is to find a subinterval of I_x of a particularly nice form, a so-called binary interval. This interval is completely specified by a binary string z. It is of the form $J_z = \{0.\omega : \omega \in \{0,1\}^* \text{ and } \omega_{1:l(z)} = z\}$. Notice that the length of this interval is $2^{-l(z)}$.

Thus let $J_z \subseteq I_x$ be the longest binary interval contained in I_x. For uniqueness, we take J_z to be the leftmost such interval. It is easy to see that J_z is of length at least $1/4$ that of I_x. Our encoding of x will be z, the left endpoint of J_z, which has length at most $-\log \mu(x) + 2$.

Let $g(\cdot,\cdot)$ be a recursive function witnessing the recursiveness of μ. Given z which is the encoding for x, we design the following test which accepts y if and only if $x = y$. By cycling through all y we can then find and print x.

Given input y, we choose k such that $|\mu(y) - g(y,k)| < \mu(y)/16$ and

$$\left| \sum_{y'<y} g(y',k) - \mu(y') \right| < \mu(y)/16. \tag{2.2}$$

Clearly we can find such a k. We accept y if and only if

$$\sum_{y'<y} g(y',k) < 0.z + 2^{-l(z)-1} \text{ and } \sum_{y'\leq y} g(y',k) > 0.z + 2^{-l(z)-1}.$$

Notice the reason that the we need the factor of 1/16 is because we are only assured that the length of the binary interval corresponding to z is $\mu(x)/4$.

We now deal with the complication that the semi-probability distribution μ is not recursive but only enumerable. Let $g(\cdot,\cdot)$ be a recursive function witnessing the enumerability of μ. That is, $g(x,k) \leq g(x,k+1)$ and $\lim_{k\to\infty} g(x,k) = \mu(x)$. Let $g'(x,k)$ be such that

$$g'(x,k) = \max\{2^{-\ell} : 2^{-\ell} \leq g(x,k)\}.$$

Now let $g''(x,k)$ be the partial recursive function which enumerates the range of $g'(x,k)$ without repetition. We now see that $g''(x,k)$ is not too far from being a semi-probability distribution:

$$\begin{aligned} \sum_{x,k} g''(x,k) &= \sum_x \sum_k g''(x,k) \\ &\leq \sum_x 2\mu(x) \leq 2. \end{aligned}$$

We can thus apply the above coding construction to $g''(x,k)/2$, chopping off intervals $I_{x,k}$ of length $g''(x,k)/2$ and finding the leftmost binary interval contained in each $I_{x,k}$. As $g''(x,k)$ is recursive, constructing these intervals can be done recursively. For some k it holds that $g''(x,k) \geq \mu(x)/2$, and thus for this k the length of $I_{x,k}$ will be at least $\mu(x)/4$. Thus there is a binary interval $\Gamma_{x,k} \subseteq I_{x,k}$ with length at least $\mu(x)/16$. We encode x by giving the left endpoint of $\Gamma_{x,k}$ which has length $-\log \mu(x) + 4$, and a constant number of bits to describe the above procedure. □

This theorem has many interesting implications. Perhaps most importantly, it says that the distribution $2^{-K(x)}$ dominates any other enumerable semi-probability distribution in the sense that for any enumerable probability distribution P there is a positive constant c_P such that $2^{-K(x)} \geq c_P P(x)$, for all x. In other words,

$2^{-K(x)}$ gives at least as much probability to each x as any other enumerable probability distribution, up to a constant multiplicative factor. Furthermore, notice that the distribution $2^{-K(x)}$ is itself enumerable—define the recursive function $g(x, k)$ which dovetails the running of programs, running program i for j steps where $i + j = k$. If p is a shortest program which halts with output x at step k, then the output of $g(x, k)$ is $2^{-l(p)}$. This function is recursive, nondecreasing, and converges to $2^{-K(x)}$. As $2^{-K(x)}$ dominates all enumerable semi-probability distributions and is itself an enumerable semi-probability distribution, we will say that it is universal for the class of enumerable semi-probability distributions.

The universal a priori probability distribution $Q(x)$ is

$$Q(x) = \sum_{p:U(p)=x} 2^{-l(p)},$$

where U is a Kolmogorov minimal element for self-delimiting Turing machines. The reason that this distribution is called the a priori probability is that if we know nothing then one natural way to generate a number is to flip a random program p and run the machine on p and see what it outputs. $Q(x)$ tells us the probability that we get x in this process. Another application of Theorem 2.3.7 tells us that $Q(x)$ and $2^{-K(x)}$ agree up to a constant factor.

2.3.8. Theorem (Levin). *There is a constant c such that for every x*

$$2^{-K(x)} = \sum_{p:U(p)=x} 2^{-l(p)},$$

where equality holds up to a multiplicative factor c.

Proof: It is clear that $2^{-K(x)} \leq \sum_{p:U(p)=x} 2^{-l(p)}$ as the latter sum in particular includes a shortest program for x of length $K(x)$.

For the other direction, we observe that $Q(x) = \sum_{p:U(p)=x} 2^{-l(p)}$ is an enumerable semi-probability distribution and apply Theorem 2.3.7. To see that $Q(x)$ is enumerable: at stage k we approximate $Q(x)$ by running all programs $1 \leq j \leq k$ for i many steps, where $j+i = k$, and tallying the contributions to $\sum_{p:U(p)=x} 2^{-l(p)}$ from any of these programs which halt with output x. □

This theorem has the pleasing implication that if a string has many long programs, then it must also have a short one.

2.3.4 Fourth pillar: symmetry of information

One of the most beautiful theorems in Kolmogorov complexity is the principle of symmetry of information. Roughly speaking, this theorem says the information contained in x about y is equal to the information contained in y about x, up to

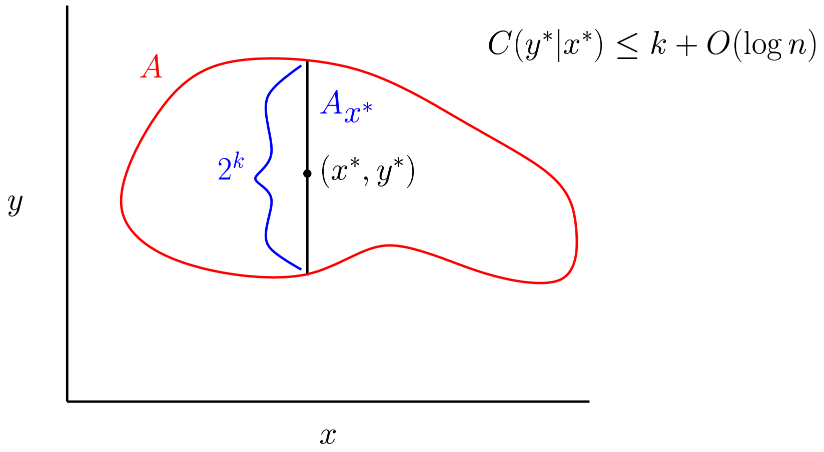

Figure 2.1: The "line" of 2^k many y with $C(y \,|\, x^*) \leq m$.

a small additive factor. This theorem has seen numerous applications in diverse areas of theoretical computer science, often where the problem does not suggest any use of Kolmogorov complexity at all. A prime example of such an application is the result of Jiang, Seiferas, and Vitányi [JSV97] who use symmetry of information to show that a Turing machine with one tape and two heads can do things that a two tape machine with one head on each tape cannot. This solved a longstanding open problem, and no alternative proof is known.

We now look at this principle in more detail.

2.3.9. Theorem. *For all $x, y \in \{0,1\}^n$, $C(x,y) = C(x) + C(y \,|\, x) + O(\log n)$.*

Proof: It is easy to see that $C(x,y) \leq C(x) + C(y \,|\, x) + O(\log n)$. Given a program for x and a program for y given x, and a way to tell the programs apart, we can first run the program for x to produce x, and then run the program for y given x and output the pair.

The other direction is more interesting. Fix $x^*, y^* \in \{0,1\}^n$, and say that $C(x^*, y^*) = m$. Consider the set of strings $A = \{(x,y) : C(x,y) \leq m\}$. This set is recursively enumerable given m— we dovetail running of all programs of length $\leq m$; if the pair (x,y) is in this set then eventually one of these programs will halt having printed (x,y). Also notice that the size of this set is less than 2^{m+1} by counting the number of short programs. As shown in Figure 2.1 consider the vertical line through x^* intersecting the set A. That is, consider the set $A_{x^*} = \{y : C(x^*, y) \leq m\}$. Notice that y^* is an element of this set, and this set is recursively enumerable given x^* and m. Therefore, by Theorem 2.3.3, we have $C(y^* \,|\, x^*) \leq \log |A_{x^*}| + O(\log n)$. Let k be such that $2^k \leq |A_{x^*}| < 2^{k+1}$.

Now consider the set $B_k = \{x : \exists^{\geq 2^k} y : C(x,y) \leq m\}$. This is the set of x with "long lines" in A, see Figure 2.2. Notice that x^* is a member of this set, that this set is recursively enumerable given m and k, and that the size of this set is less

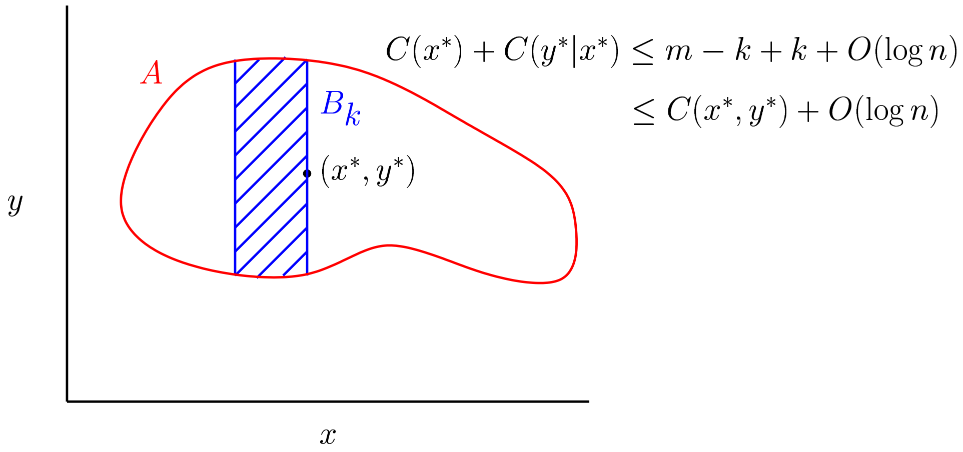
$C(x^*) + C(y^*|x^*) \leq m - k + k + O(\log n)$
$\leq C(x^*, y^*) + O(\log n)$
A
B_k
(x^*, y^*)
y
x

Figure 2.2: The set of x which have “long lines” is small

than 2^{m-k+1} since the size of A is less than 2^{m+1}. Thus applying Theorem 2.3.3 again we have $C(x^*) \leq m - k + 1 + O(\log n)$.

Putting the two together, we now have $C(x^*) + C(y^* \mid x^*) \leq m + O(\log n) \leq C(x^*, y^*) + O(\log n)$. □

2.3.5 Resource Bounded Kolmogorov Complexity

The four pillars of Kolmogorov complexity have had numerous applications in complexity theory. This is perhaps best illustrated by an example. Take the second pillar of Kolmogorov complexity, the language compression theorem. We have seen how the language compression theorem can be used to show BPP $\subseteq$ P/poly. This shows that randomness can be simulated deterministically with nonuniformity or advice. The use of Kolmogorov complexity in this application is in some sense overkill— the “bad” random strings, those which lead to a wrong answer, are actually in a small set which is decidable in polynomial time, far below the recursively enumerable sets which the language compression theorem can handle. If we could somehow take advantage of this, we could potentially get a much stronger result.

This is exactly the approach taken by Sipser in a beautiful paper showing that BPP is in the polynomial hierarchy [Sip83]. He looks at a version of polynomial time Kolmogorov complexity and shows a language compression theorem for languages in P with respect to this measure. He then shows that one can find a string with high complexity with respect to this measure in the polynomial hierarchy; then we can run the BPP algorithm using this high complexity string, which must give the right answer. We will discuss Sipser's result in more detail in Chapter 4.

Sipser's result is the main motivating factor for the results in the first part of this thesis. We look for resource-bounded versions of all the four pillars, in the hopes that that such theorems will lead to even more applications of the four pillars in complexity theory. Before we state in more detail the results we obtain, we lay the groundwork for the theory of resource bounded Kolmogorov complexity. As in the resource unbounded case, our first order of business in resource bounded Kolmogorov complexity will be to establish the existence of a Kolmogorov minimal element.

For technical reasons we will always make use of nice time bounds.

2.3.10. Definition (Nice time bounds). *We will call a function $t : \mathbb{N} \to \mathbb{N}$ a* nice time bound *if there is a Turing machine T which on input n halts with output $t(n)$ in exactly $t(n)$ many steps.*

Let $\phi^{t(n)}(p)$ be the result, if any, of the computation of ϕ on input p after $t(n)$ time steps. We define time-bounded Kolmogorov complexity as follows:

2.3.11. Definition. *Let $t(n) \geq n$ be a nice time bound, and ϕ a partial recursive function. For x, y with $n = l(\langle x, y\rangle)$ we define*

$$C_\phi^{t(n)}(x \mid y) = \min_p \left(\{ l(p) : \phi^{t(n)}(\langle p, y\rangle) = x \} \cup \{\infty\} \right).$$

As usual we set $C_\phi^{t(n)}(x) = C_\phi^{t(n)}(x \mid \varepsilon)$. It is important to remember that the running time is a function of the length of $\langle x, y\rangle$ and not the length of the input program p.

For a nice time bound $t(n) \geq n$ we wish to construct a Kolmogorov minimal element for the set $\{\phi_1^t, \phi_2^t, \phi_3^t, \ldots\}$, where $\phi_1, \phi_2, \phi_3, \ldots$ is the same recursive enumeration of partial recursive functions used above. We try to construct a Kolmogorov element for this class of functions in the same way as before: we construct ϕ such that on input $\langle i, p, y\rangle$,

$$\phi(\langle i, p, y\rangle) = \phi_i(\langle p, y\rangle).$$

Now, however, we would also like ϕ to produce the output of $\phi_i^{t(n)}(\langle p, y\rangle)$ in $t(n)$ steps. While we are not able to do this, we can show that ϕ is able to simulate ϕ_i without too much blowup in time.

There are two sources of overhead which the machine ϕ encounters in simulating the computation of ϕ_i. First, the machine ϕ must generate the description of ϕ_i from its index i via the recursive enumeration. This could take a long time, but this time only depends on i and is independent of x, y. Secondly, the machine ϕ has some fixed number k of worktapes. In our enumeration, however, there are machines which have an arbitrary number of worktapes. Thus this machine ϕ_i could have $100k$ worktapes. Luckily, a simulation due to Hennie and Stearns shows that a Turing machine with 2 worktapes can simulate the workings of a

Turing machine with an arbitrary number of worktapes with a $c \log t$ multiplicative factor blowup in time [HS66]. Thus we obtain the following weaker version of a minimal element for the set $\mathcal{F}^t$ as the function is not actually itself a member of $\mathcal{F}^t$ but rather some slightly larger class.

2.3.12. Theorem. *There is a partial recursive function ϕ such that for any partial recursive function ψ there are constants c_1, c_2, c_3 such that*

$$C_\phi^{c_1 t \log t + c_2}(x \mid y) \leq C_\psi^t(x \mid y) + c_3.$$

2.3.6 Time-bounded prefix-free complexity

It turns out it is easier to define time bounded prefix-free complexity in terms of self-delimiting Turing machines than in terms of prefix-free partial recursive functions. Let $M_1, M_2, M_3, \ldots$ be our enumeration of self-delimiting Turing machines, and let $M_i^{t(n)}(p)$ be the result, if any, of M_i on input p after $t(n)$ steps. As before, we define a self-delimiting machine M which on input $\langle i, y, p \rangle$ first decodes the index i, which can be done as i is presented in a self-delimited way, and then simulates the running of $M_i(\langle y, p \rangle)$. As before, by the simulation of Hennie and Stearns [HS66] we obtain that there are constants c_1, c_2, c_3 such that

$$C_M^{c_1 t \log t + c_2}(x \mid y) \leq C_{M_i}^t(x \mid y) + c_3$$

for all x, y.

We can also define time bounded prefix-free complexity via partial recursive prefix-free functions. In the time bounded case, however, the correspondence between this definition and that by self-delimiting Turing machines is not as tight as in the time unbounded case. We first define what it means for a function to be time t prefix-free.

2.3.13. Definition (time bounded prefix-free functions). *Let $t(n)$ be a nice time bound. We say that a Turing machine T is time $t(n)$ prefix-free if whenever $T(p)$ halts in $t(n)$ steps and $T(q)$ halts in $t(n)$ steps then p is not a proper prefix of q. Notice that a T can be time t bounded prefix-free and not prefix-free.*

Now we wish to find a minimal element for the set of time bounded prefix-free partial recursive functions. To do this we need a more efficient enumeration than that described for prefix-free partial functions. Fortunately this is made easier as we know in advance that we only have to run each machine for $t(n)$ time steps. Given a partial recursive function ϕ we modify it into a time t prefix-free function ψ as follows:

- Step 0: Initially, ψ is undefined for all inputs. Run ϕ on the empty input for t time steps. If ϕ halts then set $\psi(\varepsilon) = \phi(\varepsilon)$ and the construction of ψ is finished. Otherwise, $\psi(\varepsilon)$ remains undefined and we move on to step 1.

- Step k: In turn run ϕ on all programs of length k for t time steps. If ϕ halts in t time steps on a program p and has not in a previous stage halted on a proper prefix of p, then we set $\psi(p) = \phi(p)$. Otherwise, if ϕ has halted on a proper prefix of p then $\psi(p)$ remains undefined. Go to step $k + 1$.

In this way, ψ will be a time t prefix-free function, and if ϕ was time t prefix-free then $\psi = \phi$. Furthermore, this construction is fairly efficient in the sense that given ϕ, to evaluate ψ on a program p of length m, we need only run ϕ for t time steps on each of the $m + 1$ prefixes of p. This fact allows us, with some overhead in time, to construct a Kolmogorov minimal element for the set of time t prefix-free functions. As usual we set $\psi(\langle i, p, y\rangle) = \psi_i^t(\langle p, y\rangle)$, where ψ_i is the i^{th} function in our enumeration of time t prefix free functions. We need to check how long it takes ψ to evaluate $\psi_i^t(\langle p, y\rangle)$. By the simulation of Hennie and Stearns, ψ can simulate the running of t steps of ψ_i in time $ct \log t$. To evaluate $\psi_i(p)$, we simulate the running of ϕ_i on all of the at most $2n$ prefixes of p (as p is of length less than $2n$) to see if any of them halt in t steps. If any of them halt, then we know that $\phi_i(\langle p, y\rangle)$ is undefined. Otherwise, if none of these halt, then we output $\phi_i(\langle p, y\rangle)$. This whole simulation will take at most $cnt \log t$ steps, for some constant c. Thus we obtain a somewhat weaker theorem about the existence of a minimal element for time bounded prefix free functions.

2.3.14. Theorem. *There is a function ψ such that for any time t prefix function ψ_i there are constants c_1, c_2, c_3 such that*

$$\mathrm{C}_{\psi}^{c_1 nt \log t + c_2}(x \mid y) \leq C_{\psi_i}^{t}(x \mid y) + c_3.$$

for any x, y with $n = l(\langle x, y\rangle)$.

Let us now see the equivalence between these two definitions. Let the definition in terms of time t prefix-free functions be K_1 and the definition in terms of self-delimiting Turing machines be K_2. How are these quantities related?

First of all, notice that a time t bounded self-delimiting Turing machine computes a prefix-free function, and thus appears somewhere in our enumeration $\psi_1, \psi_2, \psi_3, \ldots$ of time t prefix-free functions. Thus our minimal element for prefix-free partial functions implies

$$K_1^{ctn \log t}(x \mid y) \leq K_2^t(x \mid y) + c.$$

Now for the other direction. We show how to simulate the running of a prefix-free function on a self-delimiting machine. Say we wish to simulate a machine T on input p. We do the following. Let $p_0 = \varepsilon, p_1, p_2, \ldots, p_m = p$ be the $m + 1$ prefixes of p. In sequence we run T on p_i for t time steps. If any of these programs halt for $i < m$ then we do not output anything. Otherwise, if none of these programs halt, then we output the output of $T(p)$. In this way we can simulate on a self-delimiting machine the running of any prefix-free function, and in particular, the universal prefix-free function constructed above. This implies

$$K_2^{ctn \log t}(x \mid y) \leq K_1^t(x \mid y) + c.$$

2.4 Summary of our results

In the first part of this thesis, we will look for analogues of the four pillars of Kolmogorov complexity in the resource bounded setting. Our first observation is a simple one: adding a time bound cannot make Kolmogorov complexity decrease. That is, $C^t(x \mid y) \geq C(x \mid y)$ for any time bound t. This means that the first pillar of Kolmogorov complexity holds unchanged in the resource-bounded case.

For the other three pillars, however, the resource bounded situation becomes quite interesting and all sorts of difficulties arise which we did not face in the unbounded setting. In fact, one benefit of looking at these theorems under the finer lens of the resource bounded setting is that we also obtain a better understanding of the essence of these theorems in the resource unbounded case. In particular, our results will show the fundamental role that compression plays in all of the pillars. The language compression theorem and the source compression theorem are clearly results about compression of different types of sets; we will see that the principle of symmetry of information is also fundamentally a compression result, which in some sense uses both compression of sets and probability distributions.

We will start in Chapter 4 by discussing the Language Compression Theorem in the resource bounded case. Our main lemma, the Compression Lemma Lemma 4.3.1, uses the theory of hardness vs. randomness tradeoffs to show that a string with high resource bounded Kolmogorov complexity can be used to generate a pseudorandom distribution. This lemma will be used several times, for both positive and negative results, throughout the first part of the thesis.

With the compression lemma in hand we go looking for resources needed for a language compression theorem. We show lower bounds that with polynomial time, even with randomness, no analogue of the language compression theorem exists. Thus we continue our search armed with more resources, and finally see that with nondeterminism we can prove an analog of the language compression theorem which asymptotically achieves the information theoretic lower bound of $\log |A|$ bits to describe any element of A. With the power of nondeterminism and randomness we are able to show an even tighter version of the language compression theorem which differs from the information theoretic optimal only by an additive factor of $\log^3 n$.

Next, in Chapter 5 we turn our attention to a resource-bounded analog of the source compression theorem, the third pillar of Kolmogorov complexity. The analog of enumerable probability distributions in the polynomial time setting are known as samplable sources. These are probability distributions which can be efficiently simulated by a probabilistic polynomial time algorithm. Unlike the case with language compression, there do not seem to be any unlikely consequences from being able to nearly optimally compress samplable sources in deterministic polynomial time. We show, however, that such compression would imply that BPP $\neq$ EXP. This statement, while believed to be true, is also believed to be difficult to prove. Thus this result suggests to try to compress samplable sources assuming some derandomization hypothesis. Recently, Luis Antunes and Lance Fortnow [AF05] have indeed been able to show that deterministic compression of samplable sources is possible under a derandomization assumption.

We also see that our Compression lemma can be applied to the case of compression of samplable sources. This allows us to show unconditionally near optimal compression of samplable sources by compression schemes using nondeterminism and randomness.

The fourth pillar of Kolmogorov complexity, the principle of symmetry of information seems to be the most difficult pillar to find an analogue of in the resource bounded case. First of all, the standard proof of symmetry of information uses both language compression and the compression of probability distributions. Secondly, in the resource bounded case, we miss a "closure property" which is present in the resource unbounded case. Although we can show that the set of strings which have low polynomial time printing complexity are in NP, they are unlikely to be in P. In this way, we end up with an unbalanced version of symmetry of information, where more resources are used on one side of the inequality than the other.

As with nondeterminism and randomness we are able to show analogues of both language compression and compression of probability distribution, we are able to show such an unbalanced principle of symmetry of information result for so-called AM complexity. We also use our compression results to construct an oracle where symmetry of information fails in a strong way for polynomial time printing complexity and nondeterministic complexity.

Resource bounded Kolmogorov complexity is one way to bring the theory of Kolmogorov complexity closer to the real world setting of data compression. A current trend in data compression is so-called lossy compression which trades some fidelity to the original data in exchange for a shorter compressed length. In other words, when decompressed, we will not get our original data back but only something "close" to our original data. In Chapter 7 we propose a variation of Kolmogorov complexity to model lossy compression, which we call Kolmogorov complexity with error. We are again interested in the behavior of this measure with respect to the pillars. The first pillar now becomes more interesting in this setting —we give tight bounds for the maximal complexity of strings with error in both the resource bounded and unbounded cases. We also give an example which shows that symmetry of information does not in general hold for complexity with error.

Chapter 3

Complexity Classes and Kolmogorov Measures

Before we get started talking about complexity classes, we first lay out some basic notation. The length of a string x will be denoted $l(x)$. We use ε for the empty string. The cardinality of a set A is denoted $|A|$. The notation $A^{=n}$ is the set of elements of A which have length n. All logarithms are base 2.

3.1 Complexity classes

Complexity theory studies the amount of resources necessary and sufficient to solve computational problems. Common resources which are looked at are time, space (memory), randomness, and nondeterminism. One can then, of course, look at various combinations of all these resources put together. Perhaps the most basic resource is time, thus we start there.

3.1.1 Time

DTIME($f(n)$): This is the class of languages recognized by a Turing machine M whose running time on inputs of length n is bounded by $f(n)$.

P: Polynomial time.

$$\mathrm{P} = \cup_{i=1}^{\infty}\mathrm{DTIME}(n^i).$$

EXP: Exponential time.

$$\mathrm{EXP} = \cup_{i=1}^{\infty}\mathrm{DTIME}(2^{n^i}).$$

It is known from the time hierarchy theorem that P $\neq$ EXP [HS65].

3.1.2 Randomness

BPP: Bounded error probabilistic polynomial time. A language L is in this class if there is a polynomial time algorithm A and polynomial $p(\cdot)$ such that

- for all $x \in L : \Pr_{r \in \{0,1\}^{p(l(x))}}[A(x,r) = 1] \geq 2/3$
- for all $x \notin L : \Pr_{r \in \{0,1\}^{p(l(x))}}[A(x,r) = 1] \leq 1/3$

RP: Randomized polynomial time. This is a one-sided error version of BPP. A language $L \in$ RP if there is a polynomial time algorithm A and polynomial $p(\cdot)$ such that

- for all $x \in L : \Pr_{r \in \{0,1\}^{p(l(x))}}[A(x,r) = 1] \geq 2/3$
- for all $x \notin L : \Pr_{r \in \{0,1\}^{p(l(x))}}[A(x,r) = 1] = 0$

3.1.3 Nondeterminism

NP: Nondeterministic polynomial time. A language L is in this class if there is a polynomial time algorithm A and polynomial $p(\cdot)$ such that

- for all $x \in L : \Pr_{r \in \{0,1\}^{p(l(x))}}[A(x,r) = 1] > 0$
- for all $x \notin L : \Pr_{r \in \{0,1\}^{p(l(x))}}[A(x,r) = 1] = 0$

It is clear from the definitions that RP $\subseteq$ NP. A discussion of NP is not complete without mentioning NP completeness. A language L is NP-complete if $L \in$ NP and it is "as hard" as any problem in NP. More formally this means that for any language $L' \in$ NP there is a polynomial time computable function $f(\cdot)$ such that $x \in L' \iff f(x) \in L$. The archetypal NP-complete problem is satisfiability: given a formula over variables $x_1, \ldots, x_n$, is there a setting of the variables which makes the formula true?

PH: The polynomial hierarchy. Let $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$ be the set of languages which can be solved by a nondeterministic machine which has an oracle for an NP-complete problem like satisfiability. This means that on any computation path, the NP machine can write satisfiability queries on a special oracle tape, and in a single time step the oracle will provide the answer to this query. One can then define inductively $\Sigma_k^p = (\Sigma_{k-1}^p)^{\mathrm{NP}}$. The polynomial hierarchy is then $\mathrm{PH} = \cup_k \Sigma_k^p$. Sipser first showed that BPP is in the polynomial hierarchy [Sip83]. It is not known whether the polynomial hierarchy is infinite.

UP: Unique nondeterministic polynomial time. This class is like NP with the additional restriction that there is a unique witness to an element being in the language. More formally, a language $L \in$ UP if there is a polynomial time algorithm A and polynomial $p(\cdot)$ such that

- for all $x \in L$ there is exactly one $r \in \{0,1\}^{p(l(x))}$ such that $A(x,r) = 1$.
- for all $x \notin L$: $\Pr_{r \in \{0,1\}^{p(l(x))}}[A(x,r) = 1] = 0$

It is clear from the definition that UP $\subseteq$ NP. The other direction is an open question. A famous result shedding some light on the relationship between NP and UP is the Valiant–Vazirani isolation lemma [VV86]. Consider the NP complete problem SAT. Given a formula ϕ, Valiant and Vazirani give a randomized procedure which given a formula ϕ on n variables outputs formulas $\phi_1, \ldots, \phi_{2n}$ such that

- If ϕ is satisfiable, then the probability that at least one of the ϕ_i is satisfiable with exactly one satisfying assignment is greater than 1/8.
- If ϕ is not satisfiable, then every ϕ_i is not satisfiable.

$\oplus$P: Parity polynomial time. A language $L \in \oplus$P if there is a polynomial time algorithm A and a polynomial $p(\cdot)$ such that

- for all $x \in L$ there are an odd number of $r \in \{0,1\}^{p(l(x))}$ with $A(x,r) = 1$.
- for all $x \notin L$ there are an even number of $r \in \{0,1\}^{p(l(x))}$ with $A(x,r) = 1$.

The Valiant–Vazirani procedure described above implies that NP $\subseteq$ RP$^{\oplus \mathrm{P}}$.

3.1.4 Circuit Models

NC^1: A formula is a binary tree with interior nodes labelled by AND and OR gates and leaves labelled by literals—that is, a variable or its negation. The depth of a formula is the length of a longest path from leaf to root, and the size of a formula is its number of leaves. NC^1 is the class of decision problems for which instances of length n can be solved by a formula of depth $c \cdot \log n$ for some fixed constant c. A nontrivial theorem of Spira [Spi71] shows that every formula can be converted into an equivalent formula whose depth is logarithmic in its size. Thus NC^1 is equivalently the set of problems which can be solved by polynomial size formulas.

P/poly: This is the class of polynomial time algorithms which take advice. The key feature of the advice is that it depends only on the input length, and cannot be tailored to each individual instance x. More formally, a language L is in P/poly if there is a polynomial $p(\cdot)$ and language $L' \in$ P such that for every n there is an advice string s_n with $l(s_n) \leq p(n)$ and $(x, s_n) \in L'$. The set of languages computed by polynomial size circuits coincides with P/poly.

3.1.5 Other classes

AM: Arthur–Merlin games. AM combines randomness and nondeterminism, and is the favorite complexity class of this author. The name Arthur–Merlin comes from the following scenario: imagine Merlin to be an all powerful wizard and Arthur a distrustful king who is willing to believe probabilistic evidence. Arthur flips some coins and based on these outcomes issues a challenge to Merlin. Merlin then has to provide a witness which answers this challenge and such that Arthur can, in his own polynomial time way, verify that this witness does indeed answer his challenge. A language L is in AM if for every $x \in L$ Merlin can answer Arthur's challenge successfully with high probability, and for every $x \notin L$ no matter what Merlin does he cannot satisfy Arthur's challenge with high probability. We now give the definition more formally:

A language $L \in$ AM if there is a polynomial time algorithm A and polynomial $p(\cdot)$ such that

- for every $x \in L : \Pr_{r \in \{0,1\}^{p(l(x))}}[\exists y \in \{0,1\}^{p(l(x))} : A(x,y,r) = 1] \geq 2/3$
- for every $x \notin L : \Pr_{r \in \{0,1\}^{p(l(x))}}[\exists y \in \{0,1\}^{p(l(x))} : A(x,y,r) = 1] \leq 1/3$

The results of [FGM$^+$89] show that we can actually assume that the probability in the first item is one—that is AM has perfect completeness. In his original paper defining AM, Babai [Bab85] also showed that a game with a constant number of rounds between Arthur and Merlin can be simulated by a game of just two rounds.

AM has a claim to be the natural randomized version of NP in the following sense. For a complexity class $\mathcal{C}$, let

$$\text{almost-}\mathcal{C} = \{L : \Pr_R[L \in \mathcal{C}^R] = 1\}.$$

In the same way that BPP = almost-P [BG81, Kur87], it also holds that AM = almost-NP [NW94].

An important algorithmic property of AM is its ability to do "approximate lower bound counting" [Bab85]. This means that given a set $A \in P$ there is an AM algorithm which accepts with high probability if $|A^{=n}| \geq 2^{k+1}$ and rejects with high probability if $|A^{=n}| \leq 2^{k-1}$. A formal statement and proof of this result can be found in Appendix A.

SZK: Statistical Zero Knowledge. Statistical Zero Knowledge can again be seen as a game between Arthur and Merlin. This time, as Arthur and Merlin converse about whether or not a string x is in a language L, we wish that Arthur learns no information from Merlin other than the membership of x in L. This seemingly hard to capture notion is made formal in the following way. Consider the two distributions:

- The interaction of Arthur and Merlin as seen by Arthur

- The output of a probabilistic polynomial time machine not interacting with anyone on input x. This is called the simulator.

A language L has a statistical zero knowledge proof if whenever $x \in L$ there is a simulator such that the distribution of the output of the simulator and the distribution of Arthur's messages have small statistical difference.

Goldreich and Vadhan [GV99] showed a natural complete problem for SZK which we will make use of later: Given a Boolean circuit $C : \{0,1\}^n \to \{0,1\}^m$, estimate the entropy of the distribution $C(U_n)$.

3.2 Kolmogorov measures

We take the liberal view in this thesis that for every complexity class one can define an associated Kolmogorov measure. In this section, we describe the different Kolmogorov measures we will use.

3.2.1 Deterministic printing complexity

For easy reference we reproduce the definitions developed in Chapter 2 here.

For a nice time boundt satisfying $t(n) \geq n$, the conditional time bounded printing complexity is defined as

$$C^t(x \mid y) = \min_p \{l(p) : U(p, y) = x \text{ in at most } t(l(x) + l(y)) \text{ steps}\}$$

where U is a Turing machine which can efficiently simulate the running of any other Turing machine. We set $C^t(x) = C^t(x \mid \varepsilon)$ where ε denotes the empty string. For reasons of space, we only write t in the superscript, but the reader should keep in mind that the time bound is $t(l(x) + l(y))$. Thus note that the running time depends not on the length of the input p, but rather the length of the output x and the given string y. When no superscript is indicated, as in $C(x \mid y)$, we mean the above definition with no time bound restriction.

We will also use a time bounded version of prefix-free complexity.

$$K^t(x \mid y) = \min_p \{l(p) : U(p, y) = x \text{ in at most } t(l(x) + l(y)) \text{ steps}\}.$$

Here U denotes a self-delimiting Turing machine which is able to efficiently simulate the running of any other self-delimiting Turing machine.

3.2.2 Randomized printing complexity

We consider a randomized version of printing complexity, CBP. Let U be a Turing machine which can efficiently simulate the running of any other Turing machine.

3.2.1. Definition. $\mathrm{CBP}^t(x \mid y)$ is the length of a shortest program p such that

1. $\Pr_{r \in \{0,1\}^t}[U(p,y,r) \text{ outputs } x] > 2/3$
2. $U(p,y,r)$ runs in $\leq t(l(x)+l(y))$ steps for all $r \in \{0,1\}^t$

We set $\mathrm{CBP}^t(x) = \mathrm{CBP}^t(x \mid \varepsilon)$.

3.2.3 Distinguishing complexity

Sipser defined a relaxation of printing complexity called *distinguishing complexity.* The time t distinguishing complexity of x given y, denoted $\mathrm{CD}^t(x \mid y)$, is the length of a shortest program which runs in time $t(l(x)+l(y))$ and accepts only the string x. We will also use a randomized version of distinguishing complexity.

3.2.2. Definition. $\mathrm{CBPD}^t(x \mid y)$ is the length of a shortest program p such that

1. $\Pr_{r \in \{0,1\}^t}[U(p,x,y,r) = 1] > 2/3$
2. $\Pr_{r \in \{0,1\}^t}[U(p,z,y,r) = 0] > 2/3$ for all $z \neq x$
3. $U(p,z,y,r)$ runs in $\leq t(l(z)+l(y))$ steps for all $z \in \{0,1\}^*$

We set $\mathrm{CBPD}^t(x) = \mathrm{CBPD}^t(x \mid \varepsilon)$.

3.2.4 Nondeterministic printing complexity

We associate to the class NP the Kolmogorov complexity measure CN. Let U_n be a nondeterministic Turing machine able to efficiently simulate the running of any other nondeterministic machine.

3.2.3. Definition. $\mathrm{CN}^t(x \mid y)$ is defined as the length of a shortest program p such that

1. $U_n(p,y)$ has at least one accepting path
2. $U_n(p,y)$ outputs x on every accepting path
3. $U_n(p,y)$ runs in $\leq t(l(x)+l(y))$ steps.

We set $\mathrm{CN}^t(x) = \mathrm{CN}^t(x \mid \varepsilon)$.

In the literature, another version of nondeterministic complexity is defined in [BFL02], called CND. This is a distinguishing version of nondeterministic complexity, defined as follows:

3.2.4. Definition. Nondeterministic distinguishing complexity $\mathrm{CND}^t(y \mid x)$ is defined as the minimal length of a program p such that

1. $U_n(p, x, y)$ accepts,

2. $U_n(p, x, z)$ rejects for all $z \neq y$,

3. $U_n(p, x, z)$ runs in at most $t(l(x) + l(z))$ steps.

It can be seen that the measures CND and CN essentially coincide, up to additive logarithmic terms in program length and constant factor blow up in time.

3.2.5. Proposition. *For any nice time bound $t(n) \geq l(n)$ and for all x*

- $\mathrm{CND}^{t+O(l(x))}(x) \leq \mathrm{CN}^t(x)$

- $\mathrm{CN}^{t+O(l(x))}(x) \leq \mathrm{CND}^t(x) + O(\log l(x))$.

Proof: The first item is clear: given a CN program we can convert it into a CND program which at the end of any accepting path checks that the string printed on this path is equal to the input. This takes an extra amount of time at most $O(l(x))$.

To see $\mathrm{CN}^{t+O(l(x))}(x) \leq \mathrm{CND}^t(x) + O(\log l(x))$: if p is a nondeterministic distinguishing program for x, a nondeterministic machine given p and $l(x)$ can guess a string of length $l(x)$ which is accepted by p and output this string. By the nature of p, the new nondeterministic machine has at least one accepting computation path and outputs x on every accepting computation path. As $U_n(p, x)$ runs in time t, the whole procedure will take time at most $t + O(l(x))$. □

Because of this equivalence, in the sequel we will refer only to CN complexity.

3.2.5 Arthur–Merlin complexity

Let U_n be a nondeterministic Turing machine able to efficiently simulate the running of any other nondeterministic Turing machine.

3.2.6. Definition. $\mathrm{CAM}^t(y \,|\, x)$ is defined as the minimal length of a program p such that

1. $\Pr_r[U_n(p, x, y, r) \text{ accepts}] > 2/3$,

2. $\Pr_r[U_n(p, x, z, r) \text{ accepts}] < 1/3$ for all $z \neq y$,

3. $U_n(p, x, z, r)$ runs in at most $t(l(x) + l(z))$ steps.

The probabilities above are taken over all $t(l(x) + l(y))$ bit strings r.

As usual, we let $\mathrm{CAM}^t(x) = \mathrm{CAM}^t(x \,|\, \varepsilon)$.

3.2.6 Oracles

We will often use relativized versions of these complexity measures, allowing the decompression program to make queries to some set A at unit cost. We will always write the oracle as a superscript next to the time bound. Thus for example for time bounded printing complexity we have

$$C^{t,A}(x \mid y) = \min_p \{ l(p) : U^A(p,y) = x \text{ in at most } t(l(x)+l(y)) \text{ steps} \}$$

where the machine U is given access to the oracle A. The relativized versions of the other complexity measures are similarly defined.

Chapter 4

Language Compression

This chapter is based on the paper:

- H. Buhrman, T. Lee, and D. van Melkebeek. Language compression and pseudorandom generators. *Computational Complexity*, 14:247–274, 2005. Special issue of selected papers from the 19th Annual IEEE Conference on Computational Complexity.

4.1 Introduction

In this chapter, we investigate the second pillar of Kolmogorov complexity, the language compression theorem, in the resource bounded setting. Remember that in the setting of the language compression theorem, we are given a set A and wish to encode the elements of A using as few bits as possible. By simple counting, we have that some element of A must require a description of size at least $\log |A|$, which we refer to as the information theoretic limit. The language compression theorem states that if the set A is recursively enumerable, then we can achieve the information theoretic limit, up to an additive factor of $O(\log n)$, where n is the length of the strings in A.

A drawback of the language compression theorem is that while we know that every string in A can be given a description of size about $\log |A|$, we do not know how *long* it might take to recreate a string $x \in A$ from this description. In the resource bounded setting, we move closer to the problem of real world data compression in that we ask that $x \in A$ be generated from its description *efficiently*. Generally we will look for schemes which work in polynomial time. Our setting differs from the task of real world data compression in that we stay within the paradigm of Kolmogorov complexity—that is, we only require the decoding algorithm to be efficient, and place no restrictions on the encoding algorithm.

4.1.1 Language Compression Problem

It is instructive to first imagine what sort of language compression theorem should hold in the resource bounded setting. Perhaps the most direct analogy with Theorem 2.3.9 would be the following:

4.1.1. Hypothesis. *For every set $A \in \mathrm{P}$ there is a polynomial $p(\cdot)$ such that for every $x \in A^{=n}$*

$$C^p(x) \leq \log |A^{=n}| + O(\log n).$$

This hypothesis, however, is unlikely to be true as it implies NP = RP. Notice that the set of witnesses to a formula ϕ are a set in P. Thus given the hypothesis, and a formula ϕ with a unique satisfying assignment we could then find the witness to ϕ. The implication then follows by the Valiant-Vazirani isolation lemma [VV86].

In its relativized form, we know that Hypothesis 4.1.1 does not hold. We can state the following folklore theorem:

4.1.2. Theorem (Folklore). *For every time bound $t = t(n)$, and $0 \leq k < n$, there is a set $A \subseteq \{0,1\}^n$ with $|A| = 2^k$ such that for all $x \in A^{=n}$,*

$$C^{t,A}(x) \geq n - \log t - 1$$

Proof: Consider all programs p of length less than ℓ. We will run all these programs for t time steps with access to the empty oracle. That is, for every question, is x in the oracle? the answer is 'no'. During this time, these programs can in total query or print at most $t2^\ell$ strings. We take our set A of size 2^k to be disjoint from this set. This can be done as long as $2^k + t2^\ell \leq 2^n$. □

Thus it seems that if we want to prove a resource bounded language compression theorem, we need to lessen our demands. Sipser did just that in defining a relaxation of printing complexity called distinguishing complexity. He then was able to prove a form of the language compression theorem for distinguishing complexity, and used this to give the first proof that BPP is in the polynomial hierarchy.

The distinguishing complexity of a string x given advice s, denoted $\mathrm{CD}(x \mid s)$, is the length of a shortest polynomial time program which on input $\langle y, s\rangle$ accepts if and only if $y = x$. Sipser shows there is an advice string s of length polynomial in n, and a polynomial time bound $p(n)$ such that for all $x \in A^{=n}$,

$$\mathrm{CD}^{p,A}(x \mid s) \leq \log |A^{=n}| + O(\log n).$$

In fact, Sipser argues that most advice strings s of the appropriate length work for all $x \in A^{=n}$.

While this theorem is essentially optimal in terms of program length, it has the drawback of requiring a polynomial sized advice string. Buhrman, Fortnow, and Laplante [BFL02] eliminate this advice string at the expense of adding a factor of 2 to the program size.

4.1.3. Theorem (Buhrman-Fortnow-Laplante). *There is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$,*

$$\mathrm{CD}^{p,A}(x) \leq 2 \log |A^{=n}| + O(\log n).$$

Furthermore, there is a program that achieves this bound and only queries the oracle A on its input, rejecting immediately if the answer is negative.

Buhrman, Laplante, and Miltersen [BLM00] demonstrate a set A with $|A| = 2^{\Omega(n)}$ such that the factor of 2 in the description length is necessary. Thus, Theorem 4.1.3 is essentially optimal for the deterministic distinguishing version of the language compression problem. The authors of [BLM00] further ask if the factor of 2 is also necessary for the nondeterministic variant of distinguishing complexity, that is the length of a shortest nondeterministic polynomial time program which accepts $x \in A^{=n}$ and only x when given oracle access to A.

4.1.2 Our Results

We answer this question and show that the factor of 2 is not necessary. In fact, we show that we can asymptotically achieve the optimal factor of 1:

4.1.4. Theorem. *There is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$,*

$$\mathrm{CN}^{p,A}(x) \leq \log |A^{=n}| + O(\sqrt{\log |A^{=n}|} \log n).$$

Furthermore, there is a program that achieves this bound and only queries the oracle at length n, rejecting immediately on any path where an answer is negative.

This bound is a slight improvement over that given in [BLvM05], where the excess term is $O(\sqrt{\log |A^{=n}|} \log n + \log^2 n)$. The improvement comes from a better analysis of Lemma 4.3.1. The notation $\mathrm{CN}^{p,A}(x)$ in Theorem 4.1.4 refers to the length of a shortest nondeterministic program that runs in time $p(l(x))$ and, when given oracle access to A, outputs x on every accepting computation path, of which there is at least one. Note that the distinction between distinguishing complexity and printing complexity disappears in a nondeterministic context since the printing program can exploit its nondeterminism to guess the unique input accepted by the distinguishing program. In particular, CN essentially coincides with nondeterministic distinguishing complexity.

Although the bound in Theorem 4.1.4 is asymptotically optimal, the excess term of $O(\sqrt{|A^{=n}|}\log n)$ is larger than one might hope. By allowing the printing program to use randomness as well as nondeterminism, we can reduce the excess term to $O(\log^3 n)$. The printing procedure can be cast as an Arthur-Merlin game – Merlin can help Arthur to produce the correct string x with high probability by answering a question Arthur asks and, no matter what Merlin does, he cannot trick Arthur into outputting a string different from x except with small probability. We use the notation $\mathrm{CAM}^{p,A}(x)$ for the description length of a shortest such Arthur-Merlin protocol for x that runs in time $p(l(x))$ and in which Arthur has oracle access to A.

4.1.5. Theorem. *There is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$,*

$$\mathrm{CAM}^{p,A}(x) \leq \log|A^{=n}| + O(\log^3 n).$$

Furthermore, there is a program that achieves this bound and only queries the oracle at length n, rejecting immediately on any path where an answer is negative.

Furthermore, we look at the question of deterministic compression. Although now we cannot achieve something close to the information theoretic lower bound, we can show a matching upper bound to Theorem 4.1.2, up to a $O(\sqrt{|A^{=n}|}\log n)$ additive factor in description length, and a polynomial multiplicative factor in time. More precisely, we show the following:

4.1.6. Theorem. *Let A be a set. For all $x \in A^{=n}$,*

$$C^{t,A}(x) \leq \log|A^{=n}| + O(\sqrt{\log|A^{=n}|}\log n)$$

for a time bound $t = \mathrm{poly}(n)2^{n-\log|A^{=n}|}$.

Finally, we address the question whether randomness alone, without nondeterminism, is able to achieve the same compression ratio. We show that this is not the case in a strong sense. We show that there are sets A such that the length of efficient randomized generating programs for *any* string $x \in A^{=n}$ cannot even reach the same ballpark as the information theoretic lower bound of $\log|A^{=n}|$.

4.1.7. Theorem. *For all integers n, k, and t such that $0 \leq k \leq n$, there exists a set A such that $\log|A^{=n}| = k$ and for every $x \in A^{=n}$,*

$$\mathrm{CBP}^{t,A}(x) \geq n - \log|A^{=n}| - \log t - 5.$$

Here, $\mathrm{CBP}^{t,A}(x)$ denotes the minimum length of a randomized program p that runs in time t and outputs x with probability at least 2/3 when given oracle access to A.

Even for the randomized version of distinguishing of complexity, CBPD, the length of an optimal program can be up to a factor of 2 away from the information theoretic lower bound:

4.1.8. Theorem. *There exist positive constants c_1, c_2, and c_3 such that for all integers n, k, and t satisfying $k \leq c_1 n - c_2 \log t$ there exists a set A with $\log |A^{=n}| = k$ and a string $x \in A^{=n}$ such that*

$$\mathrm{CBPD}^{t,A}(x) \geq 2 \log |A^{=n}| - c_3.$$

Note that Theorem 4.1.3 implies that $\mathrm{CBPD}^{p,A}(x) \leq 2k + O(\log n)$ for some polynomial p and every $x \in A^{=n}$. Theorem 4.1.8 shows that the upper bound on CBPD implied by Theorem 4.1.3 is tight up to an additive term of $O(\log n)$.

Theorem 4.1.8 contrasts Sipser's result on CD complexity, where he showed that a random piece of information does allow us to achieve the optimal compression ratio. The distinguishing program in Sipser's result depends on the random choice, though, whereas CBPD complexity is based on a fixed program that can flip coins.

4.1.3 Our Technique

We use the hardness versus randomness tradeoffs based on the Nisan-Wigderson pseudorandom generator construction [NW94]. Given the truth-table $x \in \{0,1\}^n$ of a Boolean function, these tradeoffs define a pseudorandom generator $G_x : \{0,1\}^d \to \{0,1\}^m$ with seed length d much less than the output length m that has the following property: If the pseudorandom distribution $G_x(U_d)$ lands in a set $B \subseteq \{0,1\}^m$ with significantly different probability than the uniform distribution U_m over $\{0,1\}^m$, then x has a succinct description with respect to B and can be efficiently recovered from that description given oracle access to B [KvM02].

We apply the hardness versus randomness tradeoffs in the following way. Consider a set A and let $k = \log |A^{=n}|$. If we set B equal to the union of the range of G_x over all $x \in A^{=n}$ and set m to be slightly larger than $k + d$, then for every string x in $A^{=n}$ the pseudorandom distribution $G_x(U_d)$ lands in B with 100% certainty whereas the uniform distribution U_m lands in B with significantly smaller probability. We conclude that every $x \in A^{=n}$ can be efficiently constructed from a succinct description given oracle access to B. Moreover, the set B can be decided efficiently by a nondeterministic machine that has oracle access to A. This allows us to replace the oracle queries to B by nondeterminism and oracle queries to A, which is what we need for Theorem 4.1.4.

A similar (but simpler) reconstructive argument underlies the analysis of recent extractor constructions à la Trevisan (see [Sha02] for an excellent survey). Trevisan [Tre01] viewed the above hardness versus randomness tradeoffs as a transformation

$$\mathrm{TR} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$$

that takes two inputs, namely a truth-table $x \in \{0,1\}^n$ and a seed $y \in \{0,1\}^d$, and outputs the pseudorandom string $G_x(y)$. He observed that TR defines an extractor: For every distribution X on $\{0,1\}^n$ with sufficient minentropy k, the distribution $\mathrm{TR}(X, U_d)$ behaves very similar to the uniform distribution U_m with respect to *every* possible set B. The argument goes as follows: For a given set B, let us call a string $x \in \{0,1\}^n$ "bad" if $\mathrm{TR}(x, U_d)$ and U_m land in B with probabilities that are more than ϵ apart (where ϵ is some parameter). Since bad strings x with respect to B can be reconstructed from a short description, say of length $\ell(m, \epsilon)$, and each individual string x has probability at most 2^{-k} in a source of minentropy k, the extracted distribution lands in B with the same probability as the uniform distribution up to an error term of no more than $\epsilon + 2^{\ell(m,\epsilon)-k}$. So, in order to extract as much of the minentropy of the source as possible, one needs to minimize the description length $\ell(m, \epsilon)$. This is exactly what we need for our compression result of Theorem 4.1.4. Thus, our goals run parallel to those for designing "reconstructive" extractors that extract almost all of the minentropy of the source. We employ similar tools (such as weak designs [RRV02]) but need to deal with a few additional complications:

- In the extractor setting, it is sufficient to argue that a nonnegligible fraction of the bad strings x have a short description. In particular, the averaging argument in the standard analysis only shows that a fraction $\Theta(\epsilon/m)$ of the bad strings x has a short description. This slack in the analysis increases the error bound for the extractor only from $\epsilon + 2^{\ell(m,\epsilon)-k}$ to $\epsilon + \Theta(m/\epsilon)2^{\ell(m,\epsilon)-k}$. In our setting, however, we cannot afford to miss any string because we need a short description for *every* string in $A^{=n}$ with respect to a single oracle B.

- As a result, our descriptions need to include more information than in the extractor setting. There are two main components in the description, namely one depending on the weak designs underlying the Nisan-Wigderson pseudorandom generator, and one specifying $O(m)$ random bits used in the averaging argument. The latter component is the one which is needed in our setting but not in the extractor context. Balancing the two contributions optimally leads to the descriptions of length $m + O(\sqrt{m}\log n)$ from Theorem 4.1.4. By allowing the describing program not only the power of nondeterminism but also the power of randomness, we can, in some sense, mimic the averaging argument from the extractor setting and eliminate the need for the second component. This results in the shorter descriptions of length about m used in Theorem 4.1.5.

- Our descriptions need to be efficient – an element $x \in A^{=n}$ can be computed in polynomial time from its description and access to an oracle for B. This implies a return from the information theoretic setting to the computational setting which formed the starting point for Trevisan's and later extractors based on the reconstructive argument. Our efficiency requirements are not as strict as in the pseudorandom generator context, though, where each bit of x can be reconstructed in randomized time $(\log n)^{O(1)}$. We can afford reconstruction times of the order $n^{O(1)}$ but the process typically needs to be deterministic.

In the above argument, the Nisan-Wigderson construction may be replaced by the recent pseudorandom generators or reconstructive extractors based on multivariate polynomials [TSZS01, SU01]. However, although the latter lead to optimal hardness versus randomness tradeoffs in some sense [Uma03], they yield worse parameters than the Nisan-Wigderson construction in our context.

4.2 Preliminaries

4.2.1 Combinatorial Designs

A key ingredient of the Nisan-Wigderson generator is a collection of sets with small pairwise intersection. Following [NW94], a set system $\mathcal{S} = S_1, \ldots, S_m \subseteq [d]$ is called a (ℓ, ρ) *design* if for all i, $|S_i| = \ell$ and for all $i \neq j$ the intersection $|S_i \cap S_j| \leq \log \rho$.

Raz, Reingold, and Vadhan [RRV02] observe that a weaker property on the set system $\mathcal{S}$ suffices for the construction of the Nisan-Wigderson generator. Namely, the quantity essentially used in the analysis of the generator is a bound on $\sum_{j<i} 2^{|S_i \cap S_j|}$. Set systems with this sum bounded by $\rho \cdot (m-1)$ for all i are called (ℓ, ρ) *weak designs*. Unlike the case with designs, there exist weak designs where the universe size d does not depend on the number of sets m.

For our purposes, we need to draw another distinction in design terminology. We need a bound on $\sum_{j<i} 2^{|S_i \cap S_j|}$ in terms of i. Such designs were already constructed in [RRV02] but went unnamed. As the distinction will be important later, we give them their own name, referring to them as *uniform weak designs*.

4.2.1. Definition. Let $\mathcal{S} = (S_1, S_2, \ldots, S_m)$ be a family of sets where for all i, $S_i \subseteq [d]$ and $|S_i| = \ell$.

1. $\mathcal{S}$ is a *weak* (ℓ, ρ) *design* if $\sum_{j<i} 2^{|S_i \cap S_j|} \leq \rho \cdot (m-1)$ for all i.

2. $\mathcal{S}$ is a *uniform weak* (ℓ, ρ) *design* if $\sum_{j<i} 2^{|S_i \cap S_j|} \leq \rho \cdot (i-1)$ for all i.

Raz, Reingold, and Vadhan show the following lemma [RRV02]:

4.2.2. Lemma. *For every ℓ, m and $\rho = \rho(\ell, m) > 1$ there exists a set system $\mathcal{S} = (S_1, S_2, \ldots, S_m) \subseteq [d]$ constructible in* $\operatorname{poly}(m, d)$ *time, with either of the following properties:*

1. *$\mathcal{S}$ is a weak $(\ell, 1)$ design with $d = O(\ell^2 \log m)$.*

2. *$\mathcal{S}$ is a uniform weak (ℓ, ρ) design with $d = O(\ell^2 / \log \rho)$.*

It is worth noting that [RRV02] also show a matching lower bound, up to constant multiplicative factors, to the above construction of uniform weak designs. In particular, this means that $(\ell, 1)$ weak designs *cannot* be made uniform with the parameters given in item 1 [RRV02, Remark 19].

4.2.2 Error-Correcting Codes

The benefits of composing the Nisan-Wigderson generator with a good list-decodable code are well demonstrated [Tre01, STV01]. We will use a concatenation of a Reed-Solomon code with an Hadamard code. The combinatorial list-decoding properties of this code suffice for our main theorems; however, using additionally the fact that this code has efficient list-decoding [Sud97, KS99] allows us to prove a stronger form of our main lemma, the Compression Lemma (Lemma 4.3.1). The properties we need are summarized in the next two lemmata.

4.2.3. Lemma. *For every integer* $n \geq 0$ *and* $0 < \delta = \delta(n) \leq 1/2$*, there is a code* $\mathrm{LDC}_{n,\delta} : \{0,1\}^n \to \{0,1\}^{\bar{n}}$ *where* $\bar{n} = \mathrm{poly}(n/\delta)$ *with the following properties:*

1. $\mathrm{LDC}_{n,\delta}$ *can be evaluated in time* $\mathrm{poly}(n/\delta)$.
2. *Given any string* $\hat{y} \in \{0,1\}^{\bar{n}}$*, the list of all strings* $x \in \{0,1\}^n$ *such that* $\hat{x} = \mathrm{LDC}_{n,\delta}(x)$ *and* $\hat{y}$ *agree in at least a* $1/2 + \delta$ *fraction of the positions can be generated in time* $\mathrm{poly}(n/\delta)$.

4.2.4. Lemma. *Let* $\mathrm{LDC}_{n,\delta}$ *be as above and* $\hat{x} = \mathrm{LDC}_{n,\delta}(x)$*. For every rational* $0 < \delta = \delta(n) \leq 1/2$ *there is a time bound* $t = \mathrm{poly}(n/\delta)$ *such that for any* $\hat{y} \in \{0,1\}^{\bar{n}}$ *which agrees with* $\hat{x}$ *on a* $1/2 + \delta$ *fraction of positions,*

$$C^t(x \mid \hat{y}) \leq C^{t/2}(\delta) + O(\log(n/\delta)).$$

Proof: With $C^{t/2}(\delta) + O(\log n)$ bits we can describe δ, n, and the code $\mathrm{LDC}_{n,\delta}$ being used. Given $\hat{y}, n, \delta$, we can print the $\mathrm{poly}(n/\delta)$ codewords which agree with $\hat{y}$ on more than a $1/2 + \delta$ fraction of positions. By further specifying the index i of $\hat{x}$ in this list we can identify $\hat{x}$ and decode it to print x. This index i can be given with $O(\log(n/\delta))$ bits. As $\mathrm{LDC}_{n,\delta}$ is efficiently list decodable there is a function $t = \mathrm{poly}(n/\delta)$ bounding the running time of the above procedure. □

4.3 Compression Lemma

In this section, we translate some of the recent progress on extractors back into the pseudorandom generator setting, resulting in the main tool for our upper bound results, the Compression Lemma. We first describe the function underlying Trevisan's and later extractors, hereafter referred to as Trevisan's function.

Let $P : \{0,1\}^\ell \to \{0,1\}$ be any Boolean function, and let $\mathcal{S} = (S_1, \ldots, S_m)$ be a collection of subsets of $[d]$ where $|S_i| = \ell$. For a string $y \in \{0,1\}^d$ let $y|_{S_i}$ be the string in $\{0,1\}^\ell$ obtained by projecting y onto the coordinates specified by S_i. Then the Nisan-Wigderson generator $\mathrm{NW}_{\mathcal{S},P}$ is defined as

$$\mathrm{NW}_{\mathcal{S},P}(y) = P(y|_{S_1}) \cdots P(y|_{S_m}).$$

Given an input length n, an output length m, a quality parameter $\delta = \delta(m)$, and a design parameter $\rho = \rho(m) > 1$, we define the following function after Trevisan. Our definition will differ slightly from Trevisan's original definition in the use of weak designs instead of designs and that we use the "strong" form of Trevisan's function, where the seed of the generator is prepended to the output. It is in using this strong form of Trevisan's function that we are able to obtain a better bound in the Compression Lemma than that given in [BLvM05]. Let $\mathrm{LDC}_{n,\delta}$ be as in Lemma 4.2.3 and let $\ell = \log \bar{n}$. For $u \in \{0,1\}^n$, we view $\mathrm{LDC}(u)$ as a Boolean function $\hat{u} : \{0,1\}^\ell \to \{0,1\}$. Let $\mathcal{S}$ be a (ℓ, ρ) uniform weak design. Now we define $\mathrm{TR}_{\delta,\rho} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^d \times \{0,1\}^m$ as

$$\mathrm{TR}_{\delta,\rho}(u, y) = (y, \mathrm{NW}_{\mathcal{S},\hat{u}}(y)) = (y, \hat{u}(y|_{S_1}) \cdots \hat{u}(y|_{S_m})),$$

Note that while n, m are arbitrary, we need to take the auxiliary input length d so as to satisfy the conditions of the uniform weak design.

The property of Trevisan's function that is crucial for extractor constructions and for our results is the following lemma. It is a refinement of similar statements [NW94, Tre01, RRV02], where the result was stated for circuit size in [NW94] and (nonuniform) description size in [Tre01, RRV02]. The new feature of our version of the lemma is the combination of completeness, succinctness, and efficiency of the descriptions: *every* "bad" string with respect to B has a *very* succinct description from which it can be *efficiently* recovered given oracle access to B.

4.3.1. Lemma (Compression Lemma). *Let $B : \{0,1\}^{m+d} \to \{0,1\}$. Given $\epsilon = \epsilon(m) > 0$, let $\delta = \epsilon/m$. If*

$$|\Pr[B(\mathrm{TR}_{\delta,\rho}(u, U_d) = 1)] - \Pr[B(U_m) = 1]| \geq \epsilon$$

then for a time bound $t = \mathrm{poly}(n/\epsilon)$, we have

$$C^{t,B}(u) \leq \rho \cdot m + d + C^{t/2}(\epsilon) + O(\log(m/\epsilon)).$$

Furthermore, there is a program that achieves this bound and only makes non-adaptive queries to B.

Proof: We follow the by now standard proof [NW94, Tre01, RRV02]. The idea is to use the distinction from the uniform distribution that can be seen with B to find a bit of the output of TR which can be predicted with advantage – with this advantage we can then approximate $\hat{u}$ and give u a short printing program using Lemma 4.2.4.

Finding a bit of the output which can be predicted with advantage can be done using the hybrid argument of [GM84]. We define $m+d+1$ distributions, $D_0, \ldots, D_{m+d}$, where the first i bits of D_i are distributed according to the first i bits of $\mathrm{TR}(u, U_d)$, and the last $m+d-i$ bits of D_i are distributed according to the last $m+d-i$ bits of U_m. Thus note that D_0, and in fact D_i for all $0 \le i \le d$ are distributed as U_{m+d}, and D_{m+d} is distributed as $\mathrm{TR}(u, U_d)$. As $|\Pr[B(D_{m+d})] - \Pr[B(D_0)]| \ge \epsilon$, and all D_i are identically distributed for $0 \le i \le d$, for some $i > d$ it must be the case that $|\Pr[B(D_i)] - \Pr[B(D_{i-1})]| \ge \epsilon/m$. For convenience we remove the absolute value sign by choosing $b_0 \in \{0,1\}$ such that $\Pr[B'(D_i)] - \Pr[B'(D_{i-1})] \ge \epsilon/m$, where $B'(x) = b_0 \oplus B(x)$.

Writing the distributions D_{i-1}, D_i out explicitly, we now have:

$$\begin{aligned}\Pr_{\substack{y\\ r_i,\ldots,r_m}} &[B'(y, \hat{u}(y|_{S_1}) \cdots \hat{u}(y|_{S_{i-1}})\hat{u}(y|_{S_i}) r_{i+1} \ldots r_m)] \\ &- \Pr_{\substack{y\\ r_i,\ldots,r_m}} [B'(y, \hat{u}(y|_{S_1}) \cdots \hat{u}(y|_{S_{i-1}}) r_i r_{i+1} \ldots r_m)] > \epsilon/m\end{aligned}$$

By an averaging argument, we can fix the bits of y outside of S_i to some value y_0 and fix $r_{i+1}, \ldots, r_m$ to some values $c_{i+1}, \ldots, c_m$, while preserving the above difference. We rename $y|_{S_i}$ as x, and assume without loss of generality that $S_i = \{1, \ldots, \ell\}$. Thus we will write the seed y, which now depends only on x, as $y = xy_0$. Note that x varies uniformly over $\{0,1\}^\ell$, while $\hat{u}(y|_{S_j})$ for $j \ne i$, is now a function $\hat{u}_j$ which depends only on $|S_i \cap S_j|$ bits of x. That is,

$$\begin{aligned}\Pr_{x,b}&[B'(xy_0, \hat{u}_1(x) \cdots \hat{u}_{i-1}(x)\hat{u}(x) c_{i+1} \cdots c_m)] \\ &- \Pr_{x,b}[B'(xy_0, \hat{u}_1(x) \cdots \hat{u}_{i-1}(x) b c_{i+1} \cdots c_m)] > \epsilon/m \quad (4.1)\end{aligned}$$

Let $F(x,b) = (xy_0, \hat{u}_1(x) \cdots \hat{u}_{i-1}(x) b c_{i+1} \cdots c_m)$. Our program to approximate $\hat{u}$ does the following. On input x, b it evaluates $B'(F(x,b))$ and outputs b if this is one and $1-b$ otherwise. Let $g_b(x)$ denote the outcome of this process. We now estimate the probability that $g_b(x)$ agrees with $\hat{u}(x)$ over the choice of x, b:

$$\begin{aligned}
\Pr_{x,b}[g_b(x) = \hat{u}(x)] &= \Pr_{x,b}[g_b(x) = \hat{u}(x) | b = \hat{u}(x)] \Pr_{x,b}[b = \hat{u}(x)] \\
&+ \Pr_{x,b}[g_b(x) = \hat{u}(x) | b \neq \hat{u}(x)] \Pr_{x,b}[b \neq \hat{u}(x)] \\
&= \frac{1}{2} \Pr_{x,b}[B'(F(x,b)) = 1 | b = \hat{u}(x)] \\
&+ \frac{1}{2} \Pr_{x,b}[B'(F(x,b)) = 0 | b \neq \hat{u}(x)] \\
&= \frac{1}{2} + \frac{1}{2}\Bigg(\Pr_{x,b}[B'(F(x,b)) = 1 | b = \hat{u}(x)] \\
&- \Pr_{x,b}[B'(F(x,b)) = 1 | b \neq \hat{u}(x)] \Bigg) \\
&= \frac{1}{2} + \frac{1}{2}\Bigg(\Pr_{x}[B'(F(x,\hat{u}(x))) = 1] \\
&- \Pr_{x}[B'(F(x,1-\hat{u}(x))) = 1] \Bigg) \\
&= \frac{1}{2} + \Pr_{x,b}[B'(F(x,\hat{u}(x)) = 1] - \Pr_{x,b}[B'(F(x,b)) = 1] \\
&\geq \frac{1}{2} + \frac{\epsilon}{m}
\end{aligned}$$

By an averaging argument there is a bit $b_1 \in \{0,1\}$ such that $g_{b_1}(x)$ agrees with $\hat{u}(x)$ on at least a $1/2 + \epsilon/m$ fraction of x. The queries to B' are nonadaptive and that the running time of the approximation is $2^{O(\ell)} = \bar{n}^{O(1)} = \text{poly}(n/\epsilon)$.

To optimize the description size of the above program, it will be useful to separate its contributions into three parts:

1. the index i, the bits b_0, b_1 and $O(\log m)$ bits to make the entire description prefix free.
2. the fixed portion y_0 of the seed of size $d - \ell$.
3. the bits $c_{i+1}, \ldots, c_m$ and a description of the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$.

Clearly the first item costs $O(\log m)$ bits and the second at most d. We now focus on item three.

Each function $\hat{u}_j$ is a function on $|S_j \cap S_i|$ bits, thus we can completely specify it by its truth table with $2^{|S_i \cap S_j|}$ bits. Hence we can describe all the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$ with $\sum_{j=1}^{i-1} 2^{|S_j \cap S_i|}$ bits, by concatenating their truth functions. We can compute the set system $\mathcal{S}$ in polynomial time and given the value of i, we can compute the sizes of $|S_j \cap S_i|$ and uniquely decode each function $\hat{u}_j$. Thus as $\mathcal{S}$ is a (ℓ, ρ) uniform weak design, we can describe all the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$

in $\rho \cdot (i-1)$ bits. Now adding $m-i$ bits to describe $c_{i+1}, \ldots, c_m$ we see that item (3) will cost less than $\rho \cdot (m-1)$ bits.

Putting these three items together, we conclude there is a string $\hat{y}$ which agrees with $\hat{u}$ on a $1/2+\epsilon/m$ fraction of positions and with $C^{p,B}(\hat{y}) \leq \rho \cdot m + d + O(\log m)$. Now applying Lemma 4.2.4, we obtain the statement of the lemma. □

Substituting the uniform weak design parameters from Lemma 4.2.2 into the Compression Lemma, and optimizing with respect to ρ, we find the minimum is achieved when $\rho = 1 + \ell/\sqrt{m}$. For future reference, we record this in the following corollary.

4.3.2. Corollary. *Let B, ϵ, δ be as in Lemma 4.3.1, and let $\rho = 1 + \ell/\sqrt{m}$. If*

$$|\Pr[B(\mathrm{TR}_{\delta,\rho}(u, U_d) = 1)] - \Pr[B(U_{d+m}) = 1]| \geq \epsilon$$

then for a time bound $t = \mathrm{poly}(n/\epsilon)$, we have

$$C^{t,B}(u) \leq m + C^{t/2}(\epsilon) + O(\sqrt{m}\log(n/\epsilon)).$$

Furthermore, there is a program that achieves this bound and only makes non-adaptive queries to B.

4.4 Language Compression by Nondeterminism

In this section, we exhibit the power of nondeterminism in the context of the language compression problem. We show that Trevisan's function leads to compression close to the information-theoretic lower bound such that the compressed string can be recovered from its description by an efficient nondeterministic program that has oracle access to the containing language A.

The proof is an application of the Compression Lemma. In order to give short CN programs relative to A, it suffices to find a set B such that:

- Queries to B can be efficiently answered with an oracle for A and nondeterminism.
- For any $x \in A$, the distribution $\mathrm{TR}(x, U_d)$ lands in B with significantly different probability than the uniform distribution U_m.

Letting B be the set containing all strings of the form $\mathrm{TR}(x, e)$ where x ranges over A and e over all seeds of the appropriate length d, the first item will be satisfied. By taking the output length to be slightly larger than $\log|A| + d$, that is taking it to be "too long", we can also ensure that the second item is satisfied. We say "too long" as for this setting of m, Trevisan's function will not be an extractor for sources of min-entropy $\log|A|$, see also [TSUZ01]. We now go through the details.

4.1.4. Theorem (Restatement). *There is a polynomial $p(n)$ such that for any set A and for all $x \in A^{=n}$,*

$$\mathrm{CN}^{p,A}(x) \leq \log |A^{=n}| + O(\sqrt{\log |A^{=n}|} \log n).$$

Furthermore, there is a program that achieves this bound making nonadaptive queries to the oracle at length n.

Proof: Fix n and let $k = \log |A^{=n}|$. Let $\mathrm{TR}_{\delta,\rho} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m+d}$ be Trevisan's function with $m = k+1$. The parameters δ, ρ will be fixed later.

Define $B \subseteq \{0,1\}^{m+d}$ to be the image of $A^{=n} \times \{0,1\}^d$ under the function TR. That is, $B = \{(e,y) : \exists x \in A^{=n} : \mathrm{TR}(x,e) = (e,y)\}$.

By the choice of m it is clear that $\Pr[B(U_{m+d})] \leq 1/2$. For any element $x \in A^{=n}$, however, $\Pr[B(\mathrm{TR}(x,U_d))] = 1$. Thus applying Lemma 4.3.1 with $\epsilon = 1/2$ and $\rho = 1 + \ell/\sqrt{k}$ we obtain $C^{p,B}(x) \leq (1+\ell/\sqrt{k})(k+1) + d + O(\log n)$. As $\ell = O(\log n)$ and $d = O(\sqrt{k} \log n)$ with this choice of ρ, simplifying gives $C^{p,B}(x) \leq k + O(\sqrt{k} \log n)$.

We now show how an oracle for B can be replaced by a nondeterministic program with an oracle for A. By Lemma 4.3.1 we may assume that the queries to B are nonadaptive. It is clear the "yes" answers of the oracle B can be answered nondeterministically with an oracle for A. As the queries to B are nonadaptive, by additionally telling the program the number q of yes answers, the program can guess the q element subset of the queries which are "yes" answers and verify them. On any path where the incorrect q element subset has been guessed, at least one "yes" answer will not be verified and thus this path will reject. The description of q will only increase the program size by $O(\log n)$ bits. □

The positive use of the oracle in Theorem 4.1.4 also allows us to state the following corollary about the CN complexity of strings from an NP language.

4.4.1. Corollary. *For any set $A \in$ NP there is a polynomial $p(n)$ such that for all $x \in A^{=n}$,*

$$\mathrm{CN}^{p}(x) \leq \log |A^{=n}| + O(\sqrt{\log |A^{=n}|} \log n).$$

Proof: Consider the nondeterministic program with oracle access to A given by Theorem 4.1.4. Replace the oracle queries by guessing a membership witness and verifying it, rejecting whenever the verification fails. This gives the nondeterministic generating program we need. □

4.5 Language Compression By Nondeterminism and Randomness

In this section, we show that if we allow the decompression algorithm both the power of nondeterminism and randomness, then we can reduce the excess in the description length over the information theoretic lower bound from $O(\sqrt{k}\log n)$ to $O(\log^3 n)$.

In the proof of the Compression Lemma, we included as part of the description of $u \in A$ a setting of the random bits $c_{i+1}, \ldots, c_m$ fixed after position i. Including a setting of these bits in our description seems wasteful – the averaging argument of Lemma 4.3.1 shows that a $\theta(\epsilon/m)$ fraction of all $m - i$ bit strings would work equally well to describe u. In spite of this, we do not see how to avoid specifying them with nondeterminism only. However, if we allow randomization in our nondeterministic programs, or more precisely, if we consider Arthur-Merlin generating programs, then we can replace giving a fixed setting of random bits after position i, by sampling over a polynomial number of possible settings of these bits. The main benefit of not including these bits is that now, as in the extractor setting, we can use weak designs instead of uniform weak designs, and by the first part of Lemma 4.2.2, use designs with the optimal parameter $\rho = 1$.

One difficulty we need to address is that the number of positive oracle calls to the oracle B from Section 4.4 depends on the sequence of $m - i$ random bits $c_{i+1}, c_{i+2}, \ldots, c_m$ chosen. In the proof of Theorem 4.1.4, we included that number in the description of elements from A because this allowed us to replace oracle calls to B by oracle calls to A. When Arthur randomly picks $s(n) = \text{poly}(n)$ such sequences $r_1, r_2, \ldots, r_s$, we cannot include the number of positive oracle calls to B for every possible choice of r in the description. Instead, we include the average number of acceptances $\bar{a}$ over all possible values of r. With high probability, the total number of acceptances for the strings $r_1, \ldots, r_s$ will be within a bounded range of $s \cdot \bar{a}$. If the total number of acceptances for the strings $r_1, \ldots, r_s$ is indeed within this range, then Merlin will have limited leeway in his choice of demonstrating particular acceptances. Hence we can show that a nonnegligible fraction of $r_1, \ldots, r_s$ will give approximations to $\hat{u}$, or else we will catch Merlin cheating. The leeway Merlin has can lead to approximations of encodings $\hat{v}$ different from $\hat{u}$. However, only a small number of strings $\hat{v}$ can occur with probability comparable to that of $\hat{u}$ or better. We can thus specify $\hat{u}$ by distinguishing it from the other high likelihood encodings $\hat{v}$ with a small additional number of bits by the method of Theorem 4.1.3.

The technique of providing approximations to the average number of positive NP queries to limit Merlin's ability to cheat has been exploited before, e.g., in the context of random selfreducibility [FF93] and more recently in hardness-versus-randomness tradeoffs for nondeterministic circuits [SU01].

4.1.5. Theorem (Restatement). *There is a polynomial $p(n)$ such that for any* *set A and for all $x \in A^{=n}$,*

$$\mathrm{CAM}^{p,A}(x) \leq \log|A^{=n}| + O(\log^3 n).$$

Furthermore, there is a program which achieves this bound making nonadaptive queries to the oracle at length n.

Proof: We follow the proof of Theorem 4.1.4. Fix n and let $k = \log|A^{=n}|$. Because of the averaging argument, we will need to recover from more errors in the list decodable code and now take $\delta = \frac{1}{8m}$. We will use Trevisan's function where the underlying set system $\mathcal{S}$ is a $(\ell, 1)$ weak design. Thus let $\mathrm{TR}_{\delta,\rho} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m+d}$ be Trevisan's function with $m = k+1$.

As in the previous proof, we let the set $B \subseteq \{0,1\}^m$ be the image of $A \times \{0,1\}^d$ under the function TR. By the choice of m, for any $u \in A^{=n}$,

$$\Pr[B(\mathrm{TR}(u, U_d))] - \Pr[B(U_m)] \geq 1/2.$$

By the hybrid argument, there is an $i \in [m]$, and a setting of the bits of y outside of S_i such that

$$\Pr_{\substack{x\in\{0,1\}^{\ell},b \\ r\in\{0,1\}^{m-i}}}[B(\hat{u}_1(x)\cdots\hat{u}_{i-1}(x)\hat{u}(x)r)] - \Pr_{\substack{x\in\{0,1\}^{\ell},b \\ r\in\{0,1\}^{m-i}}}[B(\hat{u}_1(x)\cdots\hat{u}_{i-1}(x)br)] \geq \frac{1}{2m}. \tag{4.2}$$

For convenience in what follows, let $F(x, b, r) = \hat{u}_1(x)\cdots\hat{u}_{i-1}(x)br$.

Consider the following approach of approximating $\hat{u}$: On input x, pick a random $b \in \{0,1\}$ and $r \in \{0,1\}^{m-i}$ and compute $B(F(x,b,r))$; if this evaluates to 1, then output b, otherwise output $1-b$. Let $g_b(x,r)$ be the function computing this operation. As in the argument after Equation (4.1), from Equation (4.2) it follows that $\Pr_{x,b,r}[\hat{u}(x) = g_b(x,r)] \geq 1/2 + \frac{1}{2m}$. We set b to a value $b_1 \in \{0,1\}$ which preserves this prediction advantage. This value b_1 will be included as part of our description. Arthur cannot compute the function $g_{b_1}(x,r)$ himself as he needs Merlin to demonstrate witnesses for acceptance in B. We now show how to approximate the computation of $g_{b_1}(x,r)$ with an Arthur-Merlin protocol.

We say that r gives a α-approximation to $\hat{u}$ if $\Pr_x[g_{b_1}(x,r) = \hat{u}(x)] \geq \alpha$. For fixed r, we identify $g_{b_1}(x,r)$ with the string $z_{b_1,r}$ where $z_{b_1,r}$ has bit b_1 in position x if and only if $g_{b_1}(x,r) = 1$. For convenience we assume without loss of generality that $b_1 = 1$ and drop the subscript. Note that with this choice the number of ones in z_r is the number of strings x for which B accepts $\hat{u}_1(x)\cdots\hat{u}_{i-1}(x)b_1r$. With $w(z)$ we denote the number of ones in a string z.

Arthur randomly selects strings $r_1, \ldots, r_s$, each of length $m-i$, for a polynomial $s = s(n)$. Whereas in the proof of Theorem 4.1.4 we included in the description the number of acceptances by B for a particular setting of bits $c_{i+1}, \ldots, c_m$, we now include the average $\bar{a} = 2^{i-m}\sum_{x,r} g_{b_1}(x,r)$ number of acceptances over all $r \in \{0,1\}^{m-i}$. To limit Merlin's freedom in providing these acceptances, we want the number of acceptances by B over the strings $r_1, \ldots, r_s$ to be close to the expected $s \cdot \bar{a}$.

The next claim shows that with high probability the strings $r_1, \ldots, r_s$ will satisfy our requirements.

4.5.1. Claim. *For any $\gamma = \gamma(m, \bar{n}) > 0$, there exists $s = O(\bar{n}^2/\gamma^2)$ such that with probability at least $3/4$ over Arthur's choice of $r_1, \ldots, r_s$ the following two things will simultaneously happen:*

1. *A $\frac{1}{8m}$ fraction of $r_1, \ldots, r_s$ will give $\frac{1}{2} + \frac{1}{4m}$ approximations to $\hat{u}$.*
2. *The total number of acceptances by B over the strings $r_1, \ldots, r_s$ will be within γs of the expected. That is,*

$$|\sum_{j=1}^{s} w(z_{r_j}) - s\bar{a}| \leq \gamma s.$$

Proof: To lower bound the probability that both of these events happen, we upper bound the probability that each event individually does not happen and use a union bound.
Item (i): Notice that for a given r, if

$$\Pr_{x,b}[B(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x)\hat{u}(x)r)] - \Pr_{x,b}[B(\hat{u}_1(x) \cdots \hat{u}_{i-1}(x)br)] \geq \frac{1}{4m}$$

then r gives a $\frac{1}{2} + \frac{1}{4m}$-approximation of $\hat{u}$. We will say that r is bad if it does not yield a $\frac{1}{2} + \frac{1}{4m}$ approximation to $\hat{u}$. By Equation (4.2) and Markov's inequality,

$$\Pr_r[r \in \text{bad}] \leq \frac{1 - \frac{1}{2m}}{1 - \frac{1}{4m}} < 1 - \frac{1}{4m}.$$

By a Chernoff bound, for some constant $c_1 > 0$,

$$\Pr_{r_1, \ldots, r_s}[|\text{bad}| \geq \left(1 - \frac{1}{8m}\right) s] \leq \exp(-c_1 s/m^2).$$

Item (ii): By a Chernoff bound, for some constant $c_2 > 0$,

$$\Pr[|(1/s)\sum_{j=1}^{s} w(z_{r_j}) - \bar{a}| \geq \gamma] \leq 2\exp(-c_2\gamma^2 s/\bar{n}^2).$$

By taking $s = c_3\bar{n}^2/\gamma^2$ for a sufficiently large constant c_3, the probability of each item will be less than $1/8$, and the claim follows. □

After choosing the strings $r_1, \ldots, r_s$, Arthur requests Merlin to provide $s\bar{a} - s\gamma$ many witnesses for acceptances in B. Arthur verifies these witnesses and rejects if any of them fail. From the acceptances provided by Merlin, Arthur constructs the strings $z'_{r_1}, \ldots, z'_{r_s}$, where position x of the string z_{r_j} has a one if and only if Merlin provided a witness for $B(F(x, b_1, r_j)) = 1$. We now show that, with high probability, no matter which acceptances Merlin chooses to demonstrate, at least a $\frac{1}{16m}$ fraction of $z'_{r_1}, \ldots, z'_{r_s}$ will give $\frac{1}{2} + \frac{1}{8m}$ approximations of $\hat{u}$.

4.5.2. Claim. *If $r_1, \ldots, r_s$ satisfy the two conditions of the previous claim with $\gamma = \frac{\bar{n}}{256m^2}$, then for any demonstration of acceptances by Merlin at least a $\frac{1}{16m}$ fraction of $z'_{r_1}, \ldots, z'_{r_s}$ will be $\frac{1}{2} + \frac{1}{8m}$ approximations to $\hat{u}$.*

Proof: By assumption, the number of acceptances for the strings $r_1, \ldots, r_s$ is between $s\bar{a} - s\gamma$ and $s\bar{a} + s\gamma$. Since Merlin has to provide witnesses for $s\bar{a} - s\gamma$ acceptances and can never fool Arthur in providing an invalid witness, Merlin has at most $2s\gamma$ acceptances to play with. Consider them as Merlin's potential to fool Arthur.

How can z_{r_j} and z'_{r_j} differ? As Arthur verifies the witnesses provided by Merlin, wherever z'_{r_j} has a one, z_{r_j} must also have a one. Thus, if z_{r_j} and z'_{r_j} differ in t positions, then Merlin has to spend at least t units of his potential on r_j. Since Merlin's total potential is bounded by $2s\gamma$, we have that the number of r_j's such that z_{r_j} and z'_{r_j} differ in t or more positions is bounded by $2s\gamma/t$.

Under the conditions of the claim, a $\frac{1}{8m}$ fraction of the z_{r_j} are $\frac{1}{2} + \frac{1}{4m}$ approximations of $\hat{u}$. Setting $t = \frac{\bar{n}}{8m}$ and $\gamma = \frac{\bar{n}}{256m^2}$, we have that a fraction a least $\frac{1}{8m} - \frac{2\gamma}{t} = \frac{1}{16m}$ of the z'_{r_j}'s are approximations that agree with $\hat{u}$ in a fraction at least $\frac{1}{2} + \frac{1}{4m} - \frac{1}{8m} = \frac{1}{2} + \frac{1}{8m}$ of the positions. □

Now putting these claims together, and taking s to be a sufficiently large polynomial, say $s = \omega(m^4)$, we have that with probability 3/4 over Arthur's choice of $r_1, \ldots, r_s$, at least a $\frac{1}{16m}$ fraction of these settings will give $\frac{1}{2} + \frac{1}{8m}$ approximations to $\hat{u}$. A particular r_j can give a $\frac{1}{2} + \frac{1}{8m}$ approximation to at most the number of codewords that agree with it on a fraction at least $\frac{1}{2} + \frac{1}{8m}$ of the positions. By Lemma 4.2.3, this number is bounded by a polynomial $q(m)$.

Let us say that $\hat{v} \in$ LIKELY if at least a $\frac{1}{32m}$ fraction of r give a $\frac{1}{2} + \frac{1}{8m}$ approximation of $\hat{v}$. Note that the size of the set LIKELY is at most $32mq$. By Theorem 4.1.3, there is a distinguishing program p_1 of length $2\log(32mq)$ such that $p_1(\hat{u})$ accepts and $p_1(\hat{v})$ rejects for any $\hat{u} \neq \hat{v} \in$ LIKELY.

We make a list of all codewords $\hat{v}$ which agree with any of $z'_{r_1}, \ldots, z'_{r_s}$ on at least a $\frac{1}{2} + \frac{1}{8m}$ fraction of positions. We then remove all elements of this list which occur fewer than $\frac{s}{16m}$ times. With probability more than 2/3, $\hat{u}$ is on this list and all elements $\hat{v}$ on the list are in LIKELY. In that case, from the elements on the list, the distinguishing program p_1 will accept $\hat{u}$ and $\hat{u}$ only. As the list is explicit, the distinguishing program p_1 does not need to make any oracle calls.

To carry out the above procedure, we need the following information:

1. the index i, the bit b_1, the average number of acceptances $\bar{a}$ to high enough precision, and the distinguishing program p_1, and
2. a description of the functions $\hat{u}_1, \ldots, \hat{u}_{i-1}$.

Note that $O(\log n)$ bits of precision is enough to encode $\bar{a}$. Thus, the first item costs $O(\log n)$ bits. As we took $\mathcal{S}$ to be a $(\ell, 1)$ weak design, the second item costs less than $m = \log |A^{=n}| + O(\log^3 n)$ bits.

With probability more than 2/3, Merlin can make Arthur accept, and whenever Arthur accepts he produces u as output. Moreover, Arthur only queries the oracle A on strings of length $n = l(u)$, and rejects whenever an oracle query is answered negatively. □

As in the case of nondeterministic language compression, the positive use of the oracle in Theorem 4.1.5 implies that with a small $O(\log n)$ increase in description size, sets $A \in \mathrm{AM}$ can be compressed by an Arthur-Merlin protocol without reference to an oracle for the set A. It turns out, however, that a more general class of sets which we refer to as AM gap sets, can be compressed by AM protocols.

A gap set A is defined by an AM algorithm in the sense that for all $x \in A$ we are guaranteed that the algorithm will accept x with high probability. It could be the case, however, that there are strings $x \notin A$ which are also accepted with reasonable probability. The next theorem shows that, as far as compression goes, the only thing that matters is the number of strings which are accepted by the protocol with probability greater than 1/3. The compression of these AM gap sets will be very useful in Chapter 6 when we study symmetry of information.

4.5.3. Theorem. *Let A be a set and suppose there is a polynomial time bound $q(n)$, and predicate Q such that*

- *for all $u \in A^{=n}, \Pr_{r \in \{0,1\}^{q(n)}}[\exists y\; Q(u, y, r) = 1] \geq 2/3$*
- *$\|\{u \in \{0,1\}^n : \Pr_{r \in \{0,1\}^{q(n)}}[\exists y\; Q(u, y, r) = 1] > 1/3\}\| \leq 2^k$,*

and for all u, y, r the predicate $Q(u, y, r)$ can be computed in time $q(n)$. Then there is a polynomial time bound $p(n)$ such that for all $u \in A^{=n}$, we have $\mathrm{CAM}^p(u) \leq k + O(\log^3 n)$.

Proof: By amplification and the results of [FGM+89], we can transform the predicate Q into a predicate Q' taking random strings of length a polynomial $q'(n)$ and with the property

- if $u \in A^{=n}$ then $\Pr_r[\exists y\; Q'(u, y, r) = 1] = 1$

- $\|\{u : \Pr_r[\exists y\ Q'(u,y,r) = 1] \geq 2^{-n-2}\}\| \leq 2^k$

for r chosen uniformly over $\{0,1\}^{q'(n)}$. Let $L = \{u : \Pr_r[\exists y\ Q'(u,y,r) = 1] \geq 2^{-n-2}\}$.

For each $r \in \{0,1\}^{q'(n)}$ we define a subset of $\{0,1\}^{m+d}$ by

$$B_r = \{(e,z) \ : \ \exists u \in \{0,1\}^n, \exists y \ \mathrm{TR}(u,e) = (e,z) \wedge Q'(u,y,r) = 1\}$$

In the sequel we denote by $B_r(z)$ the predicate $z \in B_r$.

Clearly if $u \in A^{=n}$, then $\Pr_e[B_r(\mathrm{TR}(u,e))] = 1$, for any $r \in \{0,1\}^{q'(n)}$. Now for a randomly chosen $z \in \{0,1\}^{m+d}$ and randomly chosen $r \in \{0,1\}^{q'(n)}$, we calculate the probability that $z \in B_r$. As for a binary variable the probability of being 1 is equal to the expectation of the variable, we have

$$\Pr_{r,z}[z \in B_r] = E_{r,z}[B_r(z)].$$

By linearity of expectation, we can divide the latter into two contributions, that from elements $(e,z) \in \{0,1\}^{m+d}$ for which $\exists u \in L$ and such that $\mathrm{TR}(u,e) = (e,z)$, and those (e,z) for which this is not the case.

$$E_{r,z}[B_r(z)] = \sum_{\substack{(e,z)=\mathrm{TR}(u,e) \\ u \in L}} E[B_r(z)] + \sum_{\substack{(e,z)\neq\mathrm{TR}(u,e) \\ u \in L}} E[B_r(z)]$$

By taking $m = k+2$ the first term can be bounded by $1/4$. The second term is bounded by $2^m 2^{-n-2} \leq 1/4$. Going back to probability notation, we have for any $u \in A^{=n}$

$$\Pr_{r,e}[B_r(\mathrm{TR}(u,e))] - \Pr_{r,z}[B_r(z)] \geq 1/2.$$

It follows by the hybrid argument that there is an $i \in [m]$ and a setting of the bits of e outside of the set S_i such that

$$\Pr_{x,r,r'}[B_r(\hat{u}_1(x)\ldots\hat{u}_{i-1}(x)\hat{u}_i(x)r'] - \Pr_{x,r,r',b}[B_r(\hat{u}_1(x)\ldots\hat{u}_{i-1}(x)br')] \geq \frac{1}{2m}. \quad (4.3)$$

When the bits of e outside of S_i are fixed, all the functions $\hat{u}_i$ only depend on the bits inside of S_i, thus the variable x in the above ranges uniformly over $\|S_i\|$ bit strings.

Let $F(x,b,r') = \hat{u}_1(x)\ldots\hat{u}_{i-1}(x)br'$. Our algorithm to approximate $\hat{u}_i$ will do the following: on input x, choose uniformly at random b, r, r' and evaluate $B_r(F(x,b,r'))$; if this evaluates to 1, then output b, otherwise output $1-b$. Call the output of this algorithm $g_b(e,r,r')$. The probability that $g_b(e,r,r')$ agrees with $u_i(x)$ can be estimated as in the argument following Equation (4.1) as

$$\Pr_{x,r,r',b}[g_b(x,r,r') = \hat{u}(x)] \geq \frac{1}{2} + \frac{1}{2m}$$

The rest of the argument now proceeds with minor modifications as in the proof of Theorem 4.1.5 to show that the computation of $g_b(x,r,r')$ can be approximated by an AM algorithm. □

4.6 Deterministic Language Compression

In this section we see that the compression lemma can also be useful even if we have neither the power of nondeterminism nor randomness. Although we can no longer get close to the information theoretic lower bound of $\log |A|$, we can show a tight upper bound to the lower bound of Theorem 4.1.2, again up to an additive $O(\sqrt{\log |A|})$ factor in description size and a multiplicative polynomial factor in running time. This result again shows the wide applicability of the compression lemma.

Very little is known about language compression in the deterministic setting. The seminal paper of Goldberg and Sipser [GS91] asks the question of whether dense sets, that is sets containing a 1/poly fraction of all strings at a given length, can be optimally compressed in polynomial time. They give a randomized algorithm based on approximate arithmetic encoding which runs in polynomial time and has an expected worst case compression length $O(\log n)$ larger than the optimal.

A natural question is whether the same the same bound can be achieved deterministically. Vadhan, Trevisan, and Zuckerman [TVZ04] take up this question and show using extractors that one can deterministically achieve expected compression length slightly larger than optimal for dense sets in polynomial time. This result improves over Goldberg and Sipser in that both the encoding and the decoding can be done in deterministic polynomial time, and the expected compression length is slightly better. But the result of [TVZ04] is weaker than that of Goldberg and Sipser in that they consider expected compression length instead of worst case—in fact, in the scheme of Vadhan, Trevisan, and Zuckerman some strings are encoded with the identity coding of n bits.

Our result is again incomparable to the previous two. We achieve efficient deterministic decoding (but not also encoding as [TVZ04]) with respect to worst case description length. The main drawback to our result is that we suffer an excess term of $O(\sqrt{\log |A|})$ in our description length.

4.1.6. Theorem (Restatement). *For any set A and all $x \in A^{=n}$,*

$$C^{t,A}(x) \leq \log |A| + O(\sqrt{\log |A^{=n}|} \log n)$$

for a time bound $t = \mathrm{poly}(n) 2^{n - \log |A^{=n}|}$.

Proof: Fix n and let $k = \log |A^{=n}|$. Let $\mathrm{TR}_{\delta,\rho} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m+d}$ be Trevisan's function with $m = k + 1$. The parameters δ, ρ will be fixed later.

We wish to find a set B such that for every $x \in A$ the probability over y that $\mathrm{TR}(x, y)$ lands in B is bounded away from the probability an $m + d$ bit string chosen uniformly at random lands in B. Furthermore, given an $m + d$ bit string z we wish to be able to decide the question $z \in B$ in time about 2^{n-k}. Towards this aim, we first define two auxiliary sets B' and B''.

Let $B' = \{z \in \{0,1\}^{m+d} : \exists x \in A, y, \mathrm{TR}(x,y) = z\}$. Further let $B'' = \{z \in \{0,1\}^{m+d} :$ there are at least 2^{n-k+1} many (x,y) such that $\mathrm{TR}(x,y) = z\}$. B'' is the set of "heavy elements" which have at least 4 times the average degree. Now we set $B = B' \cup B''$. Let us check that B has the desired properties.

- By construction, $\Pr_y[\mathrm{TR}(x,y) \in B] = 1$ for any $x \in A^{=n}$

- Now we check the probability a random string lands in B. The cardinality of B' is at most $2^{k+d} = 2^{m+d-1}$. For a fixed string y, at most $1/4$ many m bit strings can have 4 times the average degree, thus $|B''| \leq 2^{m+d-2}$. In total, by a union bound, we have $|B| \leq (3/4)2^m$ which implies $\Pr_z[z \in B] \leq 3/4$.

- Now we see how to decide if a given string z is in B. The key idea is to determine the number of solutions in x of the equation $\mathrm{TR}(x,y) = z$, for fixed z (and therefore also fixed y as y is the first d bits of z). If this number of solutions is large then we know that $z \in B''$. If it is not too large, then we can afford to enumerate all the solutions x and check if any of them is in A.

  For a fixed y, Trevisan's function $\mathrm{TR}_y(x) = \mathrm{TR}(x,y)$ is a linear transformation over $GF(2)$. In fact we can view TR_y as the product of two matrices: the matrix describing the error-correcting code, and the matrix describing the restrictions of y in the design system. This latter matrix will have exactly one one in each row. To decide if $z \in B$ we first determine through Gaussian elimination if the matrix equation $\mathrm{TR}_y x = z$ has a solution over $GF(2)$, where y is the first d bits of z. This can be done in polynomial time. If it does not have a solution, then we answer "no". If it does have a solution, then there are $2^{n-\mathrm{rk}(\mathrm{TR}_y)}$ many solutions where $\mathrm{rk}(\mathrm{TR}_y)$ is the rank of the matrix TR_y over $GF(2)$.. Again $\mathrm{rk}(\mathrm{TR}_y)$ is something we can compute in polynomial time. If this value is larger than 2^{n-k+1} then we answer $z \in B$. Otherwise, we enumerate solutions to $\mathrm{TR}_y(x) = z$. For each solution we query $x \in A$ and answer $z \in B$ if ever the answer is yes. This will take time at most $\mathrm{poly}(n)2^{n-k}$. Thus the total running time will be $\mathrm{poly}(n)2^{n-k}$.

Applying the Compression Lemma with $\epsilon = 1/8$ and the optimal value of ρ as given in Corollary 4.3.2 we find that $C^{t,B}(x) \leq k + \sqrt{k}\log n$ for any $x \in A^{=n}$ and a polynomial time bound t. The theorem now follows as we as we can answer each oracle query to B in time $\mathrm{poly}(n)2^{n-k}$, and there are at most $\mathrm{poly}(n)$ many queries. □

4.7 Lower Bounds

In this section we will see some lower bounds on language compression in the resource bounded setting. All these bounds will be stated given the set as an oracle.

First, we argue that randomness barely helps to efficiently generate a string from a short description. In fact, the following result proves that there are sets A of size 2^k such that *no* string in A can be generated with probability at least $2/3$ by an efficient randomized program of size a bit less than $n - k$. Recall that achieving the information-theoretic bound would require programs of size k.

4.1.7. THEOREM. *For all integers n, k, and t such that $0 \leq k \leq n$, there exists a set A such that $\log |A^{=n}| = k$ and for every $x \in A^{=n}$,*

$$\mathrm{CBP}^{t,A}(x) \geq n - \log |A^{=n}| - \log t - 5.$$

Proof: We will argue that there are many strings x of length n that (i) are not generated with high probability by a randomized program p of small size with access to the empty oracle and (ii) have a small probability of being queried by any program p of small size that runs in time t and has access to the empty oracle. Putting 2^k such strings in the oracle A does not affect the output distribution of any of these programs p by much, so they still cannot generate any of the strings x we put in A.

Let's call a randomized program p small if its length is less than some integer ℓ which we'll determine later. Let B_i denote the set of inputs x of length n for which there exists a small program p that outputs x with probability at least $1/2$ on the empty oracle. Since every program can induce at most two elements in B_i and there are less than 2^ℓ small programs, we have that $|B_i| \leq 2^{\ell+1}$.

Consider the set of strings y such that p queries y with probability at least 2^{-s} on the empty oracle, where s is another integer we'll set later. If p runs in time t, the size of this set is bounded by $2^s t$. Let B_q denote the set of all queries y of length n that are asked with probability at least 2^{-s} by at least one small program p on the empty oracle. We have that $|B_q| \leq 2^{\ell+s} t$.

Let A be a set of 2^k strings of length n that are neither in B_i nor in B_q. Such a set exists provided

$$2^k \leq 2^n - 2^{\ell+s+1} t. \tag{4.4}$$

Now, consider any small program p with access to oracle A. Since A does not contain any string in B_q, the probability that p outputs something different on the empty oracle and on oracle A is no more than 2^{k-s}. Thus, for any string x outside of B_i, the probability that p outputs x on oracle A is less than $\frac{1}{2} + 2^{k-s}$. Setting $s = k + \log 6$ and using the fact that every string in A is outside of B_i, we have that no string in A can be generated by p with probability at least $\frac{1}{2} + \frac{1}{6} = \frac{2}{3}$ on oracle A. Setting $\ell = n - k - \log t - 5$ satisfies Equation (4.4), and thereby finishes the proof. □

In the absence of nondeterminism, the distinction between generating programs and distinguishing programs becomes relevant. Indeed, Theorem 4.1.3 implies that randomized distinguishing programs can do much better than the randomized generating programs from Theorem Theorem 4.1.7: We can realize an upper bound of roughly $2\log|A^{=n}|$ in the case of distinguishing programs, even for deterministic ones. [BLM00] proved that the factor of 2 is tight in the deterministic setting. We now extend that result to the randomized setting, i.e., we exhibit a set A that contains an exponential number 2^k of strings of length n such that at least one of these strings cannot be distinguished from the other strings in A by a randomized program of length a little bit less than $2k$ with oracle access to A.

As in [BLM00], the core of the argument is a combinatorial result on cover free set systems. A family $\mathcal{F}$ of sets is called K-cover free if for any different sets $F_0, \ldots, F_k \in \mathcal{F}$, $F_0 \not\subseteq \cup_{j=1}^{K} F_j$. The combinatorial result we use states that K-cover free families of more than K^3 sets need a universe of at least K^2 elements.

4.7.1. Lemma ([DR82]). *If $\mathcal{F}$ is a K-cover free family containing M sets over a universe of L element universe, and $M > K^3$ then $L \geq \frac{K^2 \log M}{2\log K + c}$ for some constant c.*

The connection between distinguishing programs and cover free families is the following. Recall that for a given string x and oracle A, a randomized distinguishing program accepts x with probability at least $2/3$ on oracle A, and rejects every other string with probability at least $2/3$ on oracle A. Let F_x^A denote the set of randomized programs of length less than ℓ that accept x with probability more than $1/2$ on oracle A. If every string in A has a randomized distinguishing program of size less than ℓ on oracle A, then the family $\{F_x^A : x \in A^{=n}\}$ is K-cover free for $K = |A^{=n}| - 1$.

The size of this family is only $M = K + 1$. In order to obtain a larger family, we argue that if all strings in A are of length n and Kolmogorov random with respect to the other strings in A, then no short efficient program p on input $x \in A$ has a noticeable probability of querying a string in A other than x. Thus, $p^A(x)$ and $p^{\{x\}}(x)$ behave essentially the same. Notice that $p^{\{x\}}(x)$ does not depend on A. This allows us to consider a larger set B containing $M > K^3$ strings x of length n that are Kolmogorov random with respect to the other strings in B. Assuming every subset A of B of size $2^k = K - 1$ has an efficient randomized distinguishing program of size less than ℓ when given oracle access to A, we have that the family

$$\mathcal{F} = \{F_x^{\{x\}} : x \in B\} \tag{4.5}$$

is a K-cover free family of size $M > K^3$. Lemma 4.7.1 then implies that $\ell \geq 2k - O(1)$.

We now fill in the details of the proof.

4.1.8. Theorem (Restatement). *There exist positive constants c_1, c_2, and c_3 such that for all integers n, k, and t satisfying $k \leq c_1 n - c_2 \log t$ there exists a set A with $\log |A^{=n}| = k$ and a string $x \in A^{=n}$ such that*

$$\mathrm{CBPD}^{t,A}(x) \geq 2 \log |A^{=n}| - c_3.$$

Proof: Let z be a string of length Mn such that $C(z) \geq l(z)$, where $M = 2^m$ will be determined later. Let B consist of the strings of length n obtained by chopping up z into M pieces of equal size. All M strings are guaranteed to be different as long as $m \leq n/2 - O(\log n)$; otherwise, we could obtain a short description of z by describing one of its length n segments as a copy of another one.

A key observation is the following:

4.7.2. Claim. *For every subset A of B, every $x \in A$, and every randomized program p of length less than ℓ running in time t,*

$$|\Pr[p^A(x) \textit{ accepts}] - \Pr[p^{\{x\}}(x) \textit{ accepts}]| < 1/6,$$

provided $n > \ell + c(m + \log(t+n))$, where c is some universal constant.

Proof: We will argue that every random bit sequence that leads to a different outcome for $p^A(x)$ and $p^{\{x\}}(x)$, has a short description with respect to z. Since there can only be few random bit sequences with a short description, this implies the claim.

Let us denote the outcome of p on input x, oracle O, and random bit sequence $r \in \{0,1\}^t$ by $p^O(x,r)$. If $p^A(x,r) \neq p^{\{x\}}(x,r)$, then $p^{\{x\}}(x,r)$ must query some string $y \in A$. We can describe this y with p, x, r, and an index of size $\log t$ indicating the time when the query takes place. By adding the remaining parts of z, the indices of x and y in z, and making everything prefix free, we obtain a description of $\langle z, r\rangle$. This shows

$$C(\langle z, r\rangle) \leq l(z) + l(r) - n + \ell + 2m + O(\log(t+n)).$$

Symmetry of information tells us that

$$C(\langle z, r\rangle) \geq C(z) + C(r \mid z) - O(m + \log(t+n)).$$

Since $C(z) = l(z)$, we conclude that

$$C(r \mid z) \leq l(r) - n + \ell + O(m + \log(t+n)).$$

We can make the fraction of random bit strings r that have such a short description less than 1/6 by choosing $n > \ell + c(m + \log(t+n))$ for some sufficiently large constant c. The claim follows. □

Now, suppose that for every subset A of B of size 2^k, every string $x \in A$ satisfies $\mathrm{CBPD}^{t,A}(x) < \ell$, where k, t, and ℓ are some integers. Then the family $\mathcal{F}$ defined by Equation (4.5) is K-cover free for $K = 2^k - 1$. Indeed, consider any subset A of B containing the 2^k different strings $x_0, x_1, \ldots, x_K$ from B. Let p be a randomized program of length less than ℓ that runs in time t, such that $p^A(x_0)$ accepts with probability at least $2/3$, and $p^A(x_i)$ rejects with probability at least $2/3$ for $1 \leq i \leq K$. Claim 4.7.2 implies that $p \in F_{x_0}^{\{x_0\}}$ and $p \notin F_{x_i}^{\{x_i\}}$ for any $1 \leq i \leq K$. Thus, $F_{x_0}^{\{x_0\}}$ is not covered by the union of the K sets $F_{x_i}^{\{x_i\}}$, $1 \leq i \leq K$.

Since the family $\mathcal{F}$ is of size $M = 2^m$, Lemma 4.7.1 implies that $\ell \geq 2k - c_3$ for some constant c_3, provided $M > K^3$. All size conditions can be met for values of k up to $c_1 n - c_2 \log t$ for some positive constants c_1 and c_2. □

Recall that [BLM00] established the same lower bound as in Theorem 4.1.8 for CD complexity instead of CBPD complexity. They also extended their result to CD complexity with access to an oracle in NP ∩ coNP. Similar to the formulation of Theorem 4.1.8, their extension can be phrased as follows: For every robust (NP ∩ coNP) machine M, there exist constants c_1, c_2, and c_3 such that for all integers n, k, and t satisfying $k \leq c_1 n - c_2 \log t$, there exists a set A with $\log |A^{=n}| = k$ and a string $x \in A$ such that

$$\mathrm{CD}^{t,M^A}(x) \geq 2 \log |A^{=n}| - c_3.$$

The robustness condition is implicit in the proof in [BLM00]. By a robust (NP ∩ coNP) machine M, we mean an oracle machine M such that for every oracle B, M^B behaves like an (NP ∩ coNP) machine. Note, though, that Theorem 4.1.4 implies the existence of a promise-(NP ∩ coNP) machine M and a polynomial p such that for any set A and every $x \in A$,

$$\mathrm{CD}^{p,M^A}(x) \leq \log |A^{=n}| + O(\delta(n)),$$

where $\delta(n) = \sqrt{\log |A^{=n}|}\log n$.

In a similar way, we can extend Theorem 4.1.8 as follows: For every robust (AM ∩ coAM) machine M, there exist constants c_1, c_2, and c_3 such that for all integers n, k, and t satisfying $k \leq c_1 n - c_2 \log t$, there exists a set A with $\log |A^{=n}| = k$ and a string $x \in A$ such that

$$\mathrm{CD}^{t,M^A}(x) \geq 2 \log |A^{=n}| - c_3.$$

However, without the robustness requirement, Theorem 4.1.5 implies the existence of a promise-(AM ∩ coAM) machine M and a polynomial p such that for any set A and every $x \in A$,

$$\mathrm{CD}^{p,M^A}(x) \leq \log |A^{=n}| + O(\log^3 n).$$

Chapter 5

Samplable Sources and Time Limited Universal Distributions

In the last chapter, we went hunting for the resources that would allow near optimal compression of languages. We saw that such compression was possible in polynomial time using nondeterminism; we also saw lower bounds that in the oracle setting compression near the information theoretic limit is not possible by randomized printing or distinguishing programs that work in polynomial time.

Unfortunately, it is unlikely that nondeterministic compression schemes will be available for desktop computers anytime soon. In a quest for a theory of compression more practically relevant, in this chapter we ask the question: what is the *largest* class of sources which can be compressed in deterministic polynomial time? That is, instead of focusing on compressing languages and varying the resources available to the compressor, we now insist on polynomial time compression schemes and vary the *type of sources* we try to compress. As is our wont, we look for classes of sources which are computationally defined.

Samplable sources are a general class of sources proposed and investigated with this question in mind by Trevisan, Vadhan, and Zuckerman [TVZ04]. A probability distribution is called samplable if it can be efficiently simulated by a probabilistic algorithm. Samplable sources are interesting for a couple of reasons. First of all, they are a natural polynomial time analog of the enumerable probability distributions we discussed in the third pillar on compression of probability distributions. Secondly, unlike the case with languages, there does not seem to be any immediate reason why, in the setting of Kolmogorov complexity, the compression of such sources should not be possible.

Thus far, most results concerning compression of samplable sources have dealt with a more demanding model of compression than that of Kolmogorov complexity. In this model of compression, one asks for two efficiently computable functions, called encoding Enc and decoding Dec, with the property that $\mathrm{Dec}(\mathrm{Enc}(x)) = x$ and where $l(\mathrm{Enc}(x))$ is the compressed length of x. Remember that in resource-bounded Kolmogorov complexity we only demand that the decoding function be efficiently computable.

We will first review the results about compression of samplable sources in this stronger model of compression, and then discuss the implications of the compression of samplable sources in the setting of Kolmogorov complexity. We will see that the question of compression of samplable sources, as in the resource unbounded counterpart, is intimately related to time limited universal distributions. We will also see that the compression of samplable sources implies derandomization, namely that BPP $\neq$ EXP. This should not be taken as an indication that compression of samplable sources is not possible in the Kolmogorov setting as it is widely believed that BPP $\neq$ EXP. Indeed, recently Antunes and Fortnow [AF05] have shown that compression of samplable sources is possible in the Kolmogorov setting under a derandomization assumption, though they need an assumption stronger than BPP $\neq$ EXP.

5.1 Preliminaries

5.1.1. Definition. For a probability distribution X, we will let $\sup(X)$ denote the support of X, that is the set of elements x for which $\Pr[X = x] > 0$.

5.1.2. Definition. Let X_n be a probability distribution over $\{0,1\}^n$. We say that X_n is samplable if there is a probabilistic polynomial time algorithm A such that $\Pr_r[A(0^n, r) = x] = \Pr[X_n = x]$, for all $x \in \{0,1\}^n$.

The following two examples show some of the differences between languages and samplable sources.

5.1.3. Example. An interesting example of a set which is samplable in polynomial time and not known to have a polynomial time membership algorithm is the set of strings with low polynomial time printing complexity. Let $t(n)$ be a polynomial and consider the set $S_n = \{x \in \{0,1\}^{\leq n} : C^t(x) \leq m\}$. There is a sampling algorithm A whose support is the set S_n and furthermore which gives every element in S probability at least 2^{-m-1}. On input 0^n this algorithm chooses at random a program p of length at most m and then runs $U(p)$ for t time steps. If $U(p)$ halts in at most t steps with output a string of length n, then A outputs this string. Otherwise, it outputs 0^n.

5.1.4. Example. An example of a set which has a polynomial time membership algorithm but which is unlikely to be samplable is the set of satisfying assignments for a formula ϕ. Given a formula ϕ and assignment a, we can decide in polynomial time if a satisfies ϕ. If there were an algorithm to sample the witnesses of ϕ, however, we could simply run this algorithm on the all zero string to obtain a witness to the formula and verify it is satisfiable, implying P = NP.

In this chapter we will also discuss a stronger notion of compression which requires both the encoding and decoding algorithms to be efficient. Remember that in resource bounded Kolmogorov complexity we only require the decoding algorithm to be efficient. As the demands on the compressor are stronger, we relax the notion of compressibility in this model to consider *average* compressed length rather than worst-case compressed length.

5.1.5. Definition. We say that X_n is compressible to length m if there exists polynomial time computable functions $\mathrm{Enc}, \mathrm{Dec}$ such that $\mathrm{Dec}(\mathrm{Enc}(x)) = x$ for every $x \in \sup(X_n)$ and $E(l(\mathrm{Enc}(x))) \leq m$, the expected compressed length is at most m.

5.2 Time limited universal distribution

A universal recursively enumerable probability distribution is one which gives as much probability to every element x as any other recursively enumerable distribution, up to a constant multiplicative factor. As an application of the third pillar of Kolmogorov complexity, we have seen in Chapter 2 that $2^{-K(x)}$ and $Q(x) = \sum_{p:U(p)=x} 2^{-l(p)}$ are examples of universal recursively enumerable probability distributions.

Universal distributions have had many interesting applications, for example to the relationship between worst-case and average case complexity [LV92b] and to learning theory [LV92a]. A drawback of the universal distribution $2^{-K(x)}$, however, is that it is uncomputable. We are thus again motivated to look at universal distributions in the time bounded domain in the hope of finding more applications where the uncomputability of $2^{-K(x)}$ might pose a problem. Many questions still remain about time bounded universal distributions, for example it is not known if $2^{-K^t(x)}$ and $Q^t(x) = \sum_{p:U^t(p)=x} 2^{-l(p)}$ agree as in the time unbounded case. Here we see that this question is equivalent to the compression of samplable sources.

5.2.1. Theorem. *The following are equivalent:*

1. *For any samplable source X_n there is a polynomial $t(n)$ such that for all $x \in \sup(X_n)$*
$$K^t(x) \leq -\log \Pr[X_n = x] + O(\log n)$$
2. *For every polynomial $t(n)$ there is a polynomial $t'(n)$ such that $Q^t(x) \leq 2^{-K^{t'}(x)+O(\log n)}$*

Proof: $(1 \Rightarrow 2)$: Fix a polynomial $t = t(n)$. We define a sampling algorithm A over n bit strings as follows. We first choose a string p of length at most $t(n)$ under the distribution where a string of length k is given probability $2^{-k}/t(n)$. We then run $U(p)$ for $t(n)$ time steps. If $U(p)$ halts with output a string x of length n within this time, then we output x; otherwise, we output 0^n. If $\sum_{p:U^t(p)=x} 2^{-l(p)} = \gamma$, then our sampling algorithm A will output x with probability at least $\gamma/t(n)$. Notice that no program of length longer than $t(n)$ factors into this sum as in $t(n)$ the machine U would not have time to read the entire program. Now applying the hypothesis gives that there exists a polynomial $t'(n)$ such that $2^{-K^{t'}(x)} \geq \gamma 2^{-O(\log t'(n))}$.

$(2 \Rightarrow 1)$: Fix a polynomial $t(n)$ and suppose that X_n is samplable in time t by an algorithm A. Let $x \in \sup(X_n)$ with $\Pr[X_n = x] = \gamma$. With some overhead in time and space, a prefix free description of $A, 0^n, r$ forms a prefix free program for x on the universal machine U, for each r where $A(0^n, r) = x$. Thus for some polynomial $t'(n)$ we have $\sum_{p:U^{t'}(p)=x} 2^{-l(p)} \geq \gamma 2^{-O(\log n)}$. Thus by hypothesis, for some polynomial $t''(n)$ we have $K^{t''}(x) \leq -\log \gamma + O(\log t''(n))$. □

In the rest of this chapter we state our results in terms of compression of samplable sources. Of course, because of Theorem 5.2.1 the same results also hold with respect to $Q^t(x) \leq 2^{-K^{t'}(x)+O(\log t'(n)}$.

5.3 Compression of samplable sources

We first review previous negative results on the compression of samplable sources in the model stronger than Kolmogorov complexity which demands efficient encoding and decoding functions.

The following theorem is attributed to Levin by Goldberg and Sipser [GS89].

5.3.1. Theorem (Levin). *Assume that there exists a one-way function. Then there is a samplable source X with entropy n^ϵ which is not compressible to length $n-3$.*

Proof: By the results of Håstad et al. [HILL99], the existence of a one-way function implies the construction of a pseudorandom generator $G : \{0,1\}^{n^\epsilon} \rightarrow \{0,1\}^n$ with the property: for any set $C \in \mathrm{P}$

$$|\Pr[G(U_{n^\epsilon}) \in C] - \Pr[U_n \in C]| \leq 1/n.$$

Let $X_n = \cup_{r \in \{0,1\}^{n^\epsilon}} G(r)$. X_n is the support of a samplable source, as witnessed by the algorithm which on input 0^n flips a random string $r \in \{0,1\}^{n^\epsilon}$ and outputs $G(r)$. Now assume that X_n can be compressed. Then we design a set $C \in P$ which the generator does not fool, as follows:

$$x \in C \text{ if and only if } \mathrm{Dec}(\mathrm{Enc}(x)) = x \text{ and } l(\mathrm{Enc}(x)) \leq n-3.$$

As the Enc, Dec can be computed in polynomial time, the set C is decidable in polynomial time. By the assumption that X_n is compressible to length $n-3$ we have $C(x) = 1$ for every $x \in X_n$. On the other hand $|\{x : C(x) = 1\}| \leq 2^{n-2}$, and so $\Pr[U_n \in C] \leq 1/4$, thus we have a contradiction. □

5.3.2. Theorem (Trevisan, Vadhan, Zuckerman). *If* BPP $\neq$ SZK *then there is a samplable source X_n which is not compressible to length $H(X_n) + n^{1-\alpha}$ for any constant $\alpha > 0$.*

Proof:[sketch] The proof combines two ideas. The first is that the problem of estimating the entropy of a general polynomial time samplable source X_n is complete for SZK [GV99].

The second idea is that if X_n is compressible, then we can obtain a good estimate of the entropy of X_n. In particular, all that we need to estimate the entropy of X_n is the *encoding* algorithm. Say that A is a sampling algorithm for X_n. Then our algorithm to estimate the entropy of X_n on input n randomly chooses polynomially many strings $r_1, \ldots, r_m$ and outputs the average of $l(\mathrm{Enc}(A(0^n, r_i)))$. With high probability, this will give a good estimate of the entropy of X. □

Notice Theorem 5.3.2 only relied on the fact that the source X_n had an efficient *encoding* algorithm. We will now return to the setting of resource bounded Kolmogorov complexity and see the consequences of a source having only an efficient *decoding* algorithm. We will see that this implies BPP $\neq$ EXP. We first need the following lemma.

5.3.3. Lemma. *Suppose the following hold:*

- NP $\subseteq$ BPP, *and*
- *For every polynomial q there exists a polynomial p such that for all x,* $\mathrm{C}^p(x) \leq \mathrm{CBP}^q(x) + O(\log l(x))$.

Then P = NP.

Proof: By the results of Ko [Ko82], the first item implies PH $\subseteq$ BPP and NP = RP. Thus to show P=NP it suffices to derandomize RP. Let $L \in$ RP witnessed by a machine M running in polynomial time and using $m = m(n)$ random bits on an input x of length n. We shall assume that $m > n$.

By standard amplification we transform M into a machine M', which uses $m(n)^3$ random bits and for which the probability that $M'(x, r)$ rejects when $x \in L$ is less than 2^{-m^2}. As the set of random strings $r \in \{0,1\}^{m^3}$ which give the 'wrong' answer is in P given x, we can apply Theorem 4.1.4 to give that for a polynomial time bound q', $\mathrm{CN}^{q'}(r \,|\, x) \leq l(r) - m^2 + O(\delta(m))$, for any such 'bad' r, where

$\delta(m) = m^{3/2} \log m$. In particular, this means that if $\mathrm{CN}^{q'}(r) = l(r) = m^3$ then $M'(x,r)$ must accept.

We now claim that for a given length n we can construct a string of length $m' = (m(n))^3$ with high $\mathrm{CN}^{q'}$ complexity in the polynomial hierarchy. Indeed, checking that a string has maximal CN complexity can be done with a Σ_2^p oracle. Thus the lexicographically first string of length m' with maximal CN complexity, call it r^*, can be found with a Σ_3^p oracle by doing a prefix search. This means that $\mathrm{C}^{q',\Sigma_3^p}(r^*) = O(\log n)$. As the hypothesis of the theorem implies $\mathrm{PH} \subseteq \mathrm{BPP}$, by following the proof that $\mathrm{BPP}^{\mathrm{BPP}} = \mathrm{BPP}$ we obtain $\mathrm{CBP}^{q''}(r^*) = O(\log n)$ for some polynomial q''. Finally applying the second hypothesis of the theorem we have $\mathrm{C}^p(r^*) = O(\log n)$ for some polynomial p.

Thus to decide if $x \in L$ we evaluate $M'(x, U(p))$ for all programs p of length $d \log n$ for some constant d. We reject if and only if M' rejects on all these computations. U will output r^* for one of these programs p and by the above argument, if $x \in L$ then $M'(x, r^*)$ must accept. □

5.3.4. Theorem. *Suppose that for every polynomial time samplable source X_n, and every $x \in X_n$ it holds that $C^t(x) \leq -\log \Pr[X_n = x] + O(\log n)$, for some polynomial time bound $t(\cdot)$. Then* BPP $\neq$ EXP.

Proof: Suppose for contradiction that BPP = EXP. Then NP $\subseteq$ BPP, and so the first condition of Lemma 5.3.3 is satisfied. We now show that under the hypothesis of the theorem the second condition of Lemma 5.3.3 is also satisfied. This will conclude the proof as then applying Lemma 5.3.3 we have EXP $\subseteq$ BPP $\subseteq \mathrm{NP}^{\mathrm{NP}}$ = P, a contradiction to the time hierarchy theorem.

To show that the hypothesis of the theorem implies the second condition of Lemma 5.3.3, we continue with the idea in Example 5.1.3. Now we consider the set of strings which have short *randomized* programs. Consider a string $x \in \{0,1\}^n$ and suppose that $\mathrm{CBP}^t(x) = m$. Consider the following sampling algorithm A. On input 0^n, the algorithm A chooses a random string p of length $\leq m$, and a random string r of length $t(n)$, and outputs $U(p,r)$. This sampling algorithm will output x with probability at least $2^{-m-O(\log m)}$. Thus by the assumption we have $C^{t'}(x) \leq m + O(\log n)$ for some polynomial t'. □

5.4 Compression of samplable sources with nondeterminism and randomness

Finally, we mention that the techniques developed in Chapter 4 for language compression can also work to show near optimal compression of samplable source using nondeterminism and randomness.

5.4.1. Theorem. *Let X_n be a samplable source. There is a polynomial time bound $q(n)$ such that for any $x \in \sup(X_n)$,*

$$\mathrm{CAM}^q(x) \leq -\log \Pr[X_n = x] + O(\log^3 n)$$

Proof: Let $x \in \sup(X_n)$ and suppose that $2^{-k} \leq \Pr[X_n = x] < 2^{-k+1}$. By the AM approximate lower bound counting algorithm of Babai [Bab85] (see Appendix A), there is an AM algorithm which accepts any element in the support of X_n which has probability at least 2^{-k} and rejects with high probability any element from the support of X_n which has probability less than 2^{-k-1}. This algorithm leads to exactly the kind of 'AM gap sets' which we saw could be compressed in Theorem 4.5.3. The number of elements which will be accepted with probability greater than 1/3 is at most the number of elements in the support of X_n with probability greater than 2^{-k-1}. By counting, the number of such elements is at most 2^{k+1}. The theorem now follows from Theorem 4.5.3. □

Chapter 6

Symmetry of Information

This chapter is based on the paper:

- T. Lee and A. Romashchenko. Resource bounded Kolmogorov complexity revisited. *Theoretical Computer Science*, 245:2-3, pages 386–405, 2005. Special issue of selected papers from the 29th International Symposium on the Mathematical Foundations of Computer Science, 2004.

6.1 Introduction

We now turn our attention to the fourth pillar of Kolmogorov complexity, the principle of symmetry of information. Besides being one of the most beautiful theorems of Kolmogorov complexity, the principle of symmetry of information has also seen applications in diverse areas of theoretical computer science, for example in [ABK$^+$02, JSV97, VV04b]. The paper of Jiang, Seiferas, and Vitányi is particularly interesting in this respect as it uses symmetry of information to prove a result—that a Turing machine with one tape and two heads can do things that a two tape machine with one head on each tape cannot—which seemingly has nothing to do with Kolmogorov complexity, and for which no alternative proof is known.

A main motivation to studying symmetry of information in the resource bounded case is that a resource bounded symmetry of information theorem could find even wider applicability within complexity theory. This was realized early on in the theory of Kolmogorov complexity, and as early as 1967 Kolmogorov suggested time-bounded versions of symmetry of information as an interesting avenue of research. The argument of Kolmogorov-Levin [ZL70] which we saw in the introduction can be used without modification to show that symmetry of information holds for programs using exponential time or polynomial space [LM93]; for polynomial time bounds the situation becomes much more subtle. A more detailed survey and open problems can be found in Section 7.1 of [LV97].

The main contributions to the problem of polynomial time symmetry of information appear in the series of works [LM93, LW95] which show, in particular, the following:

- If P=NP then polynomial time symmetry of information holds [LW95].
- If cryptographic one-way functions exist, then polynomial time symmetry of information does not hold up to a $O(\log n)$ factor [LM93, LW95].

The intuition behind the second result is, if f is a polynomial time computable one-way function, and $f(x) = y$, then y is simple given x. On the other hand, if x is simple in polynomial time given y then this would provide a way to invert the function, by cycling through all small programs.

These works leave open several interesting questions:

- Can polynomial time symmetry of information hold up to a factor larger than $O(\log n)$? The same argument sketched above shows that if symmetry of information holds up to a factor of $\delta(n)$ then there do not exist polynomial time computable cryptographic functions which cannot be inverted in time $2^{\delta(n)}$. On the other hand, it can easily be seen that $2C(x, y) \geq C(x) + C(y \mid x)$. Could we show $(2 - \epsilon)C(x, y) \geq C(x) + C(y \mid x)$ for some $\epsilon > 0$?
- We have seen many different versions of polynomial time Kolmogorov complexity in this thesis, such as distinguishing complexity and nondeterministic complexity. Could symmetry of information hold for these measures? The connection between symmetry of information and invertibility of one-way functions shown by [LM93, LW95] seems to no longer apply in the case of distinguishing complexity or nondeterministic complexity. Now if f is a polynomial time computable one-way permutation and $f(x) = y$, then $\mathrm{CD}^{\mathrm{poly}}(x \mid y)$ is constant, as with a description of f, on input z we accept if and only if $f(z) = y$.
- Is there an assumption weaker than P=NP which implies polynomial time symmetry of information?

Addressing the first two questions, we show relativized worlds where symmetry of information fails in a strong way for $\mathrm{CD}^{\mathrm{poly}}$ and $\mathrm{CN}^{\mathrm{poly}}$ (the existence of such worlds was claimed in [BF95], though without a complete proof). On the other hand, we show that for any set $A \in \mathrm{NP}$ symmetry of information holds for most pairs of strings $\langle x, y\rangle \in A$ with respect to the measure $\mathrm{CAM}^{\mathrm{poly}}$. We also unconditionally show that $\mathrm{C}^{\mathrm{poly}}(x, y) \geq \mathrm{CAM}^{\mathrm{poly}}(x) + \mathrm{CAM}^{\mathrm{poly}}(y \mid x)$. This implies that symmetry of information holds under the condition $\mathrm{C}^{\mathrm{poly}}(x \mid y) \leq \mathrm{CAM}^{\mathrm{poly}}(x \mid y)$. We show that this statement, however, is equivalent to P=NP. The main tool in both our positive and negative results are the language compression theorems from Chapter 2.

Another interesting approach to the definition of time-bounded Kolmogorov complexity is L. Levin's Kt complexity introduced in [Lev73]. Recently D. Ronneburger proved that symmetry of information does not hold for Kt complexity in a very strong sense [Ron04].

6.2 Symmetry of Information Properties

Denote by $\mathcal{C}^{\mathrm{poly}}$ a version of polynomial time-bounded Kolmogorov complexity, which can be $\mathrm{C}^{\mathrm{poly}}$, $\mathrm{CD}^{\mathrm{poly}}$, $\mathrm{CN}^{\mathrm{poly}}$, or $\mathrm{CAM}^{\mathrm{poly}}$. To formulate the problem of symmetry of information more precisely, we isolate three associated properties. The first is the *Easy Direction of Symmetry of Information*:

$$\begin{array}{l}\text{For any polynomial } p \text{ there exists a}\\ \text{polynomial } q \text{ such that for all } x, y:\\ \mathcal{C}^{q(n)}(x,y) \leq \mathcal{C}^{p(n)}(x) + \mathcal{C}^{p(n)}(y \mid x) + O(\log(n)),\\ \text{where } n = l(x) + l(y).\end{array} \tag{EDSI}$$

It is easy to verify that (EDSI) holds for any of the above complexity measures. Next is the *Hard Direction of Symmetry of Information*:

$$\begin{array}{l}\text{For any polynomial } p \text{ there exists a}\\ \text{polynomial } q \text{ such that for all } x, y:\\ \mathcal{C}^{q(n)}(x) + \mathcal{C}^{q(n)}(y \mid x) \leq \mathcal{C}^{p(n)}(x,y) + O(\log(n)),\\ \text{where } n = l(x) + l(y).\end{array} \tag{HDSI}$$

Finally we also consider the property of *Symmetry of Mutual Information*:

$$\begin{array}{l}\text{For any polynomial } p \text{ there exists a}\\ \text{polynomial } q \text{ such that for all } x, y:\\ \mathcal{C}^{q}(x) + \mathcal{C}^{q}(y \mid x) \leq \mathcal{C}^{p}(y) + \mathcal{C}^{p}(x \mid y) + O(\log n)\end{array} \tag{SMI}$$

Notice that if both (EDSI) and (HDSI) hold for a complexity measure $\mathcal{C}$, then also (SMI) holds for $\mathcal{C}$. The converse is not necessarily true.

6.3 On CAM complexity

In this section we study symmetry of information under the CAM complexity measure. In contrast to the case of CD and CN complexity, with the power of nondeterminism and randomness we can prove some positive results, showing that some weaker versions of (HDSI) hold for CAM.

Our proof will follow the proof in the resource unbounded case as given in [ZL70]. We first review this proof to see how it can be used in our case. Let α, β be two strings such that $l(\alpha)+l(\beta) = n$, and suppose that $\mathrm{C}(\alpha, \beta) = m$. We define the set $A_{x,m} = \{y : \mathrm{C}(x, y) \leq m\}$. Notice that $|A_{x,m}| \leq 2^{m+1}$ and that given x and m the set $A_{x,m}$ is recursively enumerable. Thus as $\beta \in A_{\alpha,m}$ by the Language Compression Theorem (Theorem 2.3.3), $\mathrm{C}(\beta \,|\, \alpha) \leq \log |A_{\alpha,m}| + O(\log n)$. Let k^* be such that $2^{k^*} \leq |A_{\alpha,m}| < 2^{k^*+1}$. Then the above says that $\mathrm{C}(\beta \,|\, \alpha) \leq k^* + O(\log n)$.

Now consider the set $B_{m,k} = \{x : |A_{x,m}| \geq 2^k\}$. Notice that the size of $B_{m,k}$ is less than 2^{m-k+1}, and that $\alpha \in B_{m,k^*}$. The set $B_{m,k}$ is recursively enumerable given m, k, thus by the Language Compression Theorem, $\mathrm{C}(\alpha) \leq m - k^* + O(\log n)$. And so

$$\begin{aligned} \mathrm{C}(\alpha) + \mathrm{C}(\beta \,|\, \alpha) &\leq m - k^* + k^* + O(\log n) \\ &\leq \mathrm{C}(\alpha, \beta) + O(\log n) \end{aligned}$$

If we substitute polynomial time printing complexity in the above argument, then the set $A_{x,m}$ is in NP. Further, by the approximate lower bound counting property of AM [Bab85], detailed in Appendix A, there is an AM algorithm which accepts with high probability for $x \in B_{m,k}$ and rejects with high probability for $x \notin B_{m,k-1}$. This algorithm defines exactly the type of "AM gap set" that we saw could be compressed in Theorem 4.5.3. This allows us to show the following.

6.3.1. Theorem. *There is a polynomial $p(n)$ such that for any set $A \subset \{0,1\}^* \times \{0,1\}^*$ and all $\langle x, y\rangle \in A^{=n}$*

$$\log |A^{=n}| \geq \mathrm{CAM}^{p,A^{=n}}(x) + \mathrm{CAM}^{p,A^{=n}}(y \,|\, x) - O(\log^3 n).$$

Furthermore, if $A \in$ NP then there is a polynomial $q(n)$ such that

$$\log |A^{=n}| \geq \mathrm{CAM}^{q}(x) + \mathrm{CAM}^{q}(y \,|\, x) - O(\log^3 n).$$

Proof: Fix n and $\langle \alpha, \beta\rangle \in A^{=n}$. Denote $m = \log |A^{=n}|$ and $A_x = \{y : (x, y) \in A^{=n}\}$. Membership in the set A_x can be decided in polynomial time given x and the oracle $A^{=n}$. As $\beta \in A_\alpha$ it follows from Theorem 4.1.5 that $\mathrm{CAM}^{q,A^{=n}}(\beta \,|\, \alpha) \leq \log |A_\alpha| + O(\log^3 n)$.

Now consider the set $B_k = \{x : |A_x| \geq 2^k\}$. Let k^* be such that $2^{k^*} \leq |A_\alpha| < 2^{k^*+1}$. Then $\alpha \in B_{k^*}$. By the approximate lower bound counting property of AM [Bab85], there is a predicate Q (computable in polynomial time given the oracle $A^{=n}$) such that

- If $x \in B_k$ then $\Pr_r[\exists y Q(x, y, r) = 1] \geq 2/3$
- If $x \notin B_{k-1}$ then $\Pr_r[\exists y Q(x, y, r) = 1] \leq 1/3$

Thus if $\Pr_r[\exists y\ Q(x,y,r) = 1] > 1/3$ then $x \in B_{k-1}$. However $|A^{=n}| = 2^m$ implies that $|B_{k-1}| \leq 2^{m-k+1}$. Now by Theorem 4.5.3 we obtain $\mathrm{CAM}^{q,A^{=n}}(\alpha) \leq m - k^* + O(\log^3 n)$.

Putting the above together we have

$$\mathrm{CAM}^{q,A^{=n}}(\alpha) + \mathrm{CAM}^{q,A^{=n}}(\beta \,|\, \alpha) \leq m - k^* + k^* + O(\log^3 n) \leq m + O(\log^3 n)$$

which gives the first statement of the theorem.

To prove the "furthermore", note that approximate lower bound counting of NP sets can be done in AM [Bab85], and apply Theorem 4.5.3 to give the bound on (unrelativized) CAM complexity of NP sets. □

6.3.2. Corollary. *For any set $A \subset \{0,1\}^* \times \{0,1\}^*$ and any polynomial $p(n)$ there is a polynomial q such that for all but at most a $1/n$ fraction of $\langle x, y\rangle \in A^{=n}$,*

$$\mathrm{CAM}^{p(n),A^{=n}}(x,y) \geq \mathrm{CAM}^{q(n),A^{=n}}(x) + \mathrm{CAM}^{q(n),A^{=n}}(y \,|\, x) - O(\log^3 n).$$

Furthermore, if $A \in$ NP then

$$\mathrm{CAM}^{p(n)}(x,y) \geq \mathrm{CAM}^{q(n)}(x) + \mathrm{CAM}^{q(n)}(y \,|\, x) - O(\log^3 n).$$

Proof: For all but at most a $1/n$ fraction of $\langle x, y\rangle \in A^{=n}$ we have

$$\mathrm{CAM}^{p(n),A^{=n}}(x,y) \geq \log |A^{=n}| - O(\log n).$$

Applying Theorem 6.3.1 we get the first statement of the corollary. Applying the "furthermore" of Theorem 6.3.1 gives the furthermore here. □

6.3.3. Theorem. *For any strings $x, y \in \{0,1\}^n$, and polynomial $p(n)$ there is a polynomial $q(n)$ such that $\mathrm{C}^p(x,y) \geq \mathrm{CAM}^q(x) + \mathrm{CAM}^q(y \,|\, x) - O(\log^3 n)$.*

Proof: Fix a pair of strings $\langle \alpha, \beta\rangle$. Let $n = l(\alpha) + l(\beta)$, and suppose that $\mathrm{C}^p(\alpha,\beta) = m$. Consider the set $A = \{\langle x,y\rangle : C^p(x,y) \leq m\}$. As membership in A can be decided in nondeterministic polynomial time, we may invoke the "furthermore" of Theorem 6.3.1 to give $\log |A| \geq \mathrm{CAM}^q(\alpha) + \mathrm{CAM}^q(\beta \,|\, \alpha) - O(\log^3 n)$ for some polynomial q. On the other hand, $|A| \leq 2^{m+1}$, and the theorem is proved. □

From Theorem 6.3.3 we obtain as a corollary a result of [LW95], up to an additive $O(\log^3(n))$ factor: if P = NP then

$$\mathrm{C}^p(x,y) \geq \mathrm{C}^q(x) + \mathrm{C}^q(y \,|\, x) - O(\log^3 n).$$

More generally, the following corollary holds.

6.3.4. Corollary. *Suppose that for any polynomial $p = p(n)$ there is a polynomial $q = q(n)$ such that for any x, y, $\mathrm{C}^q(x \mid y) \le \mathrm{CAM}^p(x \mid y) + O(\log^3 n)$. Then (HDSI) holds for polynomial time printing complexity, up to an $O(\log^3 n)$ additive factor.*

We will see in Section 6.6, however, that the assumption $\mathrm{C}^q(x \mid y) \le \mathrm{CAM}^p(x \mid y) + O(\log n)$ is in fact equivalent to $\mathrm{P} = \mathrm{NP}$.

6.4 On CD complexity

In this section we show a relativized world where the inequalities (SMI) and, hence, (HDSI) fail in a strong way for $\mathrm{CD}^{\mathrm{poly}}$ complexity. The proof of the next proposition follows the idea outlined in [BF95]:

6.4.1. Proposition. *There exists an oracle A and a polynomial $p(n)$ satisfying the following condition. For any $\epsilon > 0$ and large enough n there exists a pair $\langle x, y\rangle \in \{0,1\}^n \times \{0,1\}^n$ such that*

- $\mathrm{CD}^{2^{\epsilon n}, A^{=2n}}(y) > (1-\epsilon)n - O(\log n)$,
- $\mathrm{CD}^{p(n), A^{=2n}}(x) = O(1)$,
- $\mathrm{CD}^{p(n), A^{=2n}}(y \mid x) = O(1)$ *and even* $\mathrm{C}^{p(n), A^{=2n}}(y \mid x) = O(1)$,

i.e., $\mathrm{CD}^{p(n), A^{=2n}}(x) + \mathrm{CD}^{p(n), A^{=2n}}(y \mid x) \ll \mathrm{CD}^{2^{\epsilon n}, A^{=2n}}(y) + \mathrm{CD}^{2^{\epsilon n}, A^{=2n}}(x \mid y)$. *Thus, (SMI) does not hold with the oracle A.*

Proof: Fix n and choose an incompressible pair $\langle x_n, y_n\rangle \in \{0,1\}^n \times \{0,1\}^n$. Define a mapping $f_n : \mathbb{B}^n \to \mathbb{B}^n$ as follows:

- $f_n(x_n) = y_n$,
- $f_n(z) = z$ for all $z \neq x_n$.

Now we define $A^{=2n}$. At first define two auxiliary oracles B_n and C_n: let B_n contain the graph of the function f_n (on input $\langle z, i\rangle$ the oracle B_n returns the i^{th} bit of $y = f_n(z)$) and C_n contain a single string x_n (on input $z \in \{0,1\}^n$ the oracle C_n returns 1 if and only if $z = x_n$). A query to B_n consists of $(n + \log n)$ bits, and a query to C_n consists of n bits. So a query to $B_n \oplus C_n$ can be encoded as a strings of length $(n + \log n + 1)$, which is less than $2n$. Thus, we may set $A^{=2n} = B_n \oplus C_n$.

Obviously, for some polynomial $p(n)$ we have $\mathrm{CD}^{p(n), A^{=2n}}(x_n) = O(1)$ (it is enough to query C_n to distinguish x from other stings) and $\mathrm{C}^{p(n), A^{=2n}}(y_n \mid x_n) = O(1)$ (it is enough to query from B_n the value $f_n(x_n)$).

On the other hand, $\mathrm{CD}^{2^{\epsilon n},A^{=2n}}(y_n) \geq (1-\epsilon)n - O(\log n)$. Really, let s be a shortest $\mathrm{CD}^{2^{\epsilon n}}$ program for y, and assume

$$l(s) \leq (1-\epsilon)n - D\log n$$

for a large enough constant D. If this program queries at some step $t \leq 2^{\epsilon n}$ the point x_n from the oracle C_n or any point $\langle x_n, i\rangle$ from the oracle B_n, then

$$\mathrm{C}(x_n \,|\, y_n) \leq l(s) + \log t + O(\log n),$$

and

$$\mathrm{C}(x_n, y_n) \leq l(y_n) + l(s) + \log t + O(\log n) < 2n.$$

We get a contradiction, as the pair $\langle x_n, y_n\rangle$ is incompressible. Hence, s does not query any 'interesting' points from the oracle. Thus, it can work with a trivial oracle $B'_n \oplus C'_n$ (B'_n returns the i^{th} bit of z for *any* pair $\langle z, i\rangle$, and C'_n returns 0 for *any* string z). This means that

$$\mathrm{C}(y_n) \leq l(s) + O(1) \ll n,$$

and we again get a contradiction. So, we have $l(s) \geq (1-\epsilon)n - O(\log n)$. □

6.5 On CN complexity

In this section we prove that (HDSI) and (SMI) are not true for a relativized version of polynomial time bounded CN complexity. Our proof is based on the Language Compression Theorem for CN complexity, Theorem 4.1.4.

6.5.1. Theorem. *Let $m = m(n), s = s(n), t = t(n)$ be functions such that*

$$2^{s(n)} + 2^{m(n)} < 2^n$$

and

$$t(n)2^{m(n)} \leq 2^{n-3}.$$

Then there is a polynomial $p(n)$, and sets A, X such that

- $X^{=n} \subset \{0,1\}^n$, $|X^{=n}| = 2^{s(n)}$,
- $A^{=2n} \subset \{0,1\}^n \times \{0,1\}^n$,
- $|\{y \;:\; (x,y) \in A^{=2n}\}| \geq 7/8 \cdot 2^n$ *for any* $x \in X^{=n}$,
- $|\bigcup_{x \notin X} \{y \;:\; (x,y) \in A^{=2n}\}| \leq 1/8 \cdot 2^n$,

and for large enough n, for all $x \in X^{=n}$, for at least $3/4 \cdot 2^n$ strings $y \in \{0,1\}^n$ the following conditions hold: $\langle x, y\rangle \in A^{=2n}$,

$$\begin{array}{lcl} \mathrm{CN}^{p,A^{=2n}}(x\,|\,y) & \leq & s(n) + O(\delta(n)), \\ \mathrm{CN}^{t(n),A^{=2n}}(x) & \geq & m(n) - O(1), \\ \mathrm{CN}^{t(n),A^{=2n}}(y\,|\,x) & \geq & n - O(1), \end{array}$$

where $\delta(n) = \sqrt{n}\log(n)$.

Note that the term $\delta(n) = \sqrt{n}\log(n)$ comes from Theorem 4.1.4.

6.5.2. Corollary. *There exists an oracle A such that a $\mathrm{CN}^{\mathrm{poly}}$ version of (HDSI) and (SMI) do not hold. Moreover, for any $\varepsilon \in (0,1)$ there exists a polynomial p such that for any polynomial q for large enough n*

$$(2-\varepsilon)\mathrm{CN}^{p,A^{=2n}}(x,y) < \mathrm{CN}^{q,A^{=2n}}(x) + \mathrm{CN}^{q,A^{=2n}}(y\,|\,x)$$

and

$$\mathrm{CN}^{p,A^{=2n}}(y) + \mathrm{CN}^{p,A^{=2n}}(x\,|\,y) \ll \mathrm{CN}^{q,A^{=2n}}(x) + \mathrm{CN}^{q,A^{=2n}}(y\,|\,x)$$

for most $\langle x, y\rangle \in A^{=2n}$.

Proof: It follows from Theorem 6.5.1 for $s(n) = \varepsilon n/3$, $m(n) = (1-\varepsilon/3)n$, $t(n) = 2^{\varepsilon n/6}$. □

The bound $(2-\varepsilon)$ in the first inequality of Corollary 6.5.2 is tight. This can be easily seen as,

$$\mathrm{CN}^{\mathrm{poly},A^{=2n}}(x,y) \geq \mathrm{CN}^{\mathrm{poly},A^{=2n}}(x) - O(1)$$

and

$$\mathrm{CN}^{\mathrm{poly},A^{=2n}}(x,y) \geq \mathrm{CN}^{\mathrm{poly},A^{=2n}}(y\,|\,x) - O(1).$$

Hence for any oracle A

$$2\mathrm{CN}^{p,A^{=2n}}(x,y) \geq \mathrm{CN}^{q,A^{=2n}}(x) + \mathrm{CN}^{q,A^{=2n}}(y\,|\,x) - O(1).$$

Proof: (Theorem 6.5.1) Fix an integer $n > 0$. We denote by F the characteristic function of $A^{=2n}$, i.e., $F(\langle x, y\rangle) = 1$ if $\langle x, y\rangle \in A^{=2n}$ and $F(x,y) = 0$ otherwise. We define this function in a few stages: construct a sequence of functions $F_0, F_1, \ldots, F_{2^{m(n)}-1}$,

$$F_i : \{0,1\}^n \times \{0,1\}^n \to \{0, 1, \mathrm{undef}\}.$$

For $i < j$ the function F_j should be an extension of F_i, i.e.,

$$\forall\langle a, b\rangle \text{ if } F_i(a,b) \neq \mathrm{undef} \text{ then } F_j(a,b) = F_i(a,b).$$

The initial function is trivial: $F_0(a,b) = \mathrm{undef}$ for all $\langle a, b\rangle$. In the sequel we shall define F as an extension of $F_{2^{m(n)}-1}$.

Let us introduce some notation. We say that a set $B \subset \{0,1\}^n \times \{0,1\}^n$ *respects* a function F_i if

$$\begin{cases} F_i(a,b) = 1 & \Rightarrow \quad \langle a,b\rangle \in B, \\ F_i(a,b) = 0 & \Rightarrow \quad \langle a,b\rangle \notin B. \end{cases}$$

Let $s_1, \ldots, s_{2^{m(n)}-1}$ be the list of all CN-programs of length less than $m(n)$. We suppose each program s_j can access an oracle O (the oracle is not fixed in advance). Also we suppose that each s_j is clocked and runs at most $t(n)$ steps. We say that s_j is a *well defined* CN program for an oracle O if s_j^O accepts exactly one string x.

Further define F_i by induction. Let the functions $F_0, \ldots, F_{k-1}$ be already defined. We must construct a function F_k which is an extension of F_{k-1}. Consider the program s_k. There are two possibilities:

1. for any $B \subset \{0,1\}^n \times \{0,1\}^n$ that respects F_{k-1}, the program s_k is not well defined for the oracle B;

2. there exists at least one set $B \subset \{0,1\}^n \times \{0,1\}^n$ that respects F_{k-1}, and the program s_k is well defined for the oracle B.

The first case is trivial: we set $F_k(x,y) = F_{k-1}(x,y)$ for all $\langle x,y\rangle$. In the second case there exists a set B and a string x such that s_k^B accepts x in time $T(B,x)$, which is at most $t(n)$, and rejects all other strings. If there is more than one such set, we choose a set B with the minimal possible $T(B,x)$. Denote by x_k the fixed string x. Let the list of all queries of the program $s_k^B(x_k)$ to the oracle (for one of the shortest path, i.e., for an accepting path of length $T(B,x)$) be

$$\langle a_0,b_0\rangle, \langle a_1,b_1\rangle, \ldots, \langle a_r,b_r\rangle,$$

$r < t(n)$. We include all these pairs in the oracle. More precisely, define F_k as follows:

$$\begin{array}{lcll} F_k(a,b) & = & F_{k-1}(a,b) & \text{if } F_{k-1}(a,b) \neq \text{undef}, \\ F_k(a_j,b_j) & = & 1 & \text{if } \langle a_j,b_j\rangle \in B,\ j = 0,\ldots,r, \\ F_k(a_j,b_j) & = & 0 & \text{if } \langle a_j,b_j\rangle \notin B,\ j = 0,\ldots,r, \\ F_k(a,b) & = & \text{undef} & \text{if } F_{k-1}(a,b) = \text{undef and } \langle a,b\rangle \neq \langle a_j,b_j\rangle. \end{array}$$

For any set R that respects F_k, the program s_k^R accepts the string x_k in time $T(B,x)$. Note that for a time bound $t_0 \geq T(B,x)$ the CN program s_k^R may accept also a few other strings except x_k. But for any $t_0 < T(B,x)$ the program s_k^R does not accept in time t_0 any string, because we chose x_k that provides minimum to the value $T(B,x)$. Thus, if for a time bound $t_0 \leq t(n)$ the program s_k^R accepts at least one string, it must accept also x_k. In other words, it cannot *distinguish* any string except x_k.

We have described an inductive procedure, which defines the functions $F_0, \ldots,$ $F_{2^{m(n)}-1}$. At each step i we set $F_i(a,b) \neq F_{i-1}(a,b)$ for at most $t(n)$ values $\langle a,b\rangle$. Hence the function $F_{2^{m(n)}-1}$ is equal to undef for all values in $\{0,1\}^n \times \{0,1\}^n$ except for at most $t(n)2^{m(n)}$ values.

Besides we get the list L of strings x_i which can be possibly accepted by distinguishing programs s_i^R if a set R respects $F_{2^{m(n)}-1}$. This set is rather small: $|L| < 2^{m(n)}$.

Further we choose an arbitrary set

$$X^{=n} \subset \{0,1\}^n \setminus L$$

of size $2^{s(n)}$. Now define the function F as follows:

$$\begin{array}{lcll} F(x,y) & = & F_{2^{m(n)}-1}(x,y) & \text{if } F_{2^{m(n)}-1}(x,y) \neq \text{undef}, \\ F(x,y) & = & 1 & \text{if } F_{2^{m(n)}-1}(x,y) = \text{undef and } x \in X, \\ F(x,y) & = & 0 & \text{if } F_{2^{m(n)}-1}(x,y) = \text{undef and } x \notin X. \end{array}$$

The characteristic function F defines the oracle $A^{=2n}$ and the construction is finished. Note that for any $x \in X^{=n}$

$$|\{y \ : \ (x,y) \in A^{=2n}\}| \geq 7/8 \cdot 2^n,$$

and

$$|\bigcup_{x \notin X^{=n}} \{y \ : \ (x,y) \in A^{=2n}\}| < 1/8 \cdot 2^n.$$

Now fix any string $x_0 \in X$. Obviously, $\mathrm{CN}^{t(n),A^{=2n}}(x_0) \geq m(n)$ because $x_0 \notin L$. Further, there are at least

$$2^n - 2^{m(n)}t(n) - 2^{n-3} > 3/4 \cdot 2^n$$

strings y such that

- $(x_0, y) \in A^{=2n}$,
- $(x,y) \notin A^{=2n}$ for any $x \notin X^{=n}$, and
- $\mathrm{C}^{A^{=2n}}(y \mid x_0) \geq n-3$.

Denote by y_0 any of these strings. From the conditions above it follows that

- $\mathrm{CN}^{t(n),A^{=2n}}(y_0 \mid x_0) > n - O(1)$ since resource bounded complexity is not less than plain complexity;
- $\mathrm{CN}^{p(n),A^{=2n}}(x_0 \mid y_0) \leq \log|\{x \ : (x,y_0) \in A^{=2n}\}| + O(\delta(n)) \leq s(n) + O(\delta(n))$ from Theorem 4.1.4.

□

6.6 What Implies Symmetry of Information?

Is there an assumption weaker than P=NP which would imply symmetry of information? Corollary 6.3.4 shows that symmetry of information (up to a $\log^3 n$ factor) follows from the assumption:

$$\begin{array}{l}\text{For any polynomial } p \text{ there exists a}\\ \text{polynomial } q \text{ such that for all } x, y:\\ \mathrm{C}^q(x \mid y) \leq \mathrm{CAM}^p(x \mid y) + O(\log(n)),\\ \text{where } n = l(x) + l(y).\end{array} \tag{6.1}$$

It is easily seen that this property follows from P = NP. We now see that it is in fact equivalent to P = NP.

6.6.1. Theorem. *Property 6.1 implies* P = NP.

Proof: Two consequences follow from Property 6.1

- $\mathrm{C}^p(x \mid y) \leq \mathrm{CBP}^q(x \mid y) + O(\log n)$
- $\mathrm{C}^p(x \mid y) \leq \mathrm{CN}^q(x \mid y) + O(\log n)$

The second item is shown in [FK96] to imply NP = RP. This fact can be proved as follows. If ϕ if a formula with exactly one satisfying assignment a then $\mathrm{CN}^q(a \mid \phi) = O(1)$, and thus by assumption $\mathrm{C}^p(a \mid \phi) = O(\log n)$. Thus to evaluate a formula ϕ for satisfiability, we do the Valiant-Vazirani reduction [VV86] to obtain a polynomial size list of formulas $\{\phi_1, \ldots, \phi_k\}$. If the original formula was not satisfiable then none of these formulas will be satisfiable, and if the original formula was satisfiable, then with reasonable probability one of the formulas ϕ_i will have exactly one satisfying assignment. Then for each i in turn we evaluate $U^p(s, \phi_i)$ for all short programs s of length less than $c \log n$ for some constant c. If ϕ_i has one satisfying assignment, then some short program will produce this assignment, by the assumption. Thus we obtain NP = RP.

We can now apply the Lemma 5.3.3 to obtain P = NP. □

Similar to the proof of Theorem 5.3.4, we can also show that if polynomial time symmetry of information holds then BPP $\neq$ EXP.

6.6.2. Theorem. *If (SMI) holds for polynomial time printing complexity then* BPP $\neq$ EXP.

Proof: We first show that the hypothesis of the theorem implies that for every polynomial q there is a polynomial p such that for all x, $\mathrm{C}^p(x) \leq \mathrm{CBP}^q(x) + O(\log l(x))$. This means that the second hypothesis of Lemma 5.3.3 holds, and we can then apply that Lemma to obtain the statement of the theorem.

Suppose that $\mathrm{CBP}^q(x) = k$. This means there is a program p of length k such that $U(p, r) = x$ for at least $2/3$ of the strings $r \in \{0,1\}^{q(n)}$. By counting, it must be the case that $\mathrm{C}(r \mid x) \geq l(r) - O(1)$ for one of these strings r, call it r^*. Using (SMI), there is a polynomial p for which

$$\mathrm{C}^q(r^*) + \mathrm{C}^q(x \mid r^*) \geq \mathrm{C}^p(x) + \mathrm{C}^p(r^* \mid x) - O(\log n).$$

As $\mathrm{C}^q(r^*) = \mathrm{C}^p(r^* \mid x) + O(1)$ this implies $\mathrm{C}^p(x) \leq k + O(\log n)$.

We now finish the proof of the theorem. Suppose for contradiction that EXP $\subseteq$ BPP. This implies that NP $\subseteq$ BPP and thus with the above we now have both of the hypothesis of Lemma 5.3.3. Applying this lemma gives that P = NP and so EXP $\subseteq$ BPP $\subseteq$ $\mathrm{NP}^{\mathrm{NP}}$ = P, a contradiction to the time hierarchy theorem. □

We now turn to relativizations to help us find a good candidate hypothesis, weaker than P = NP, which would imply symmetry of information. As we know that symmetry of information implies the nonexistence of cryptographic one-way functions, it is natural to ask if the converse holds. This is a tantalizing hypothesis as it is known that the nonexistence of one-way functions does imply a strong compression result [Wee04, Theorem 6.3]. We show that this implication does not hold in every relativized world. That is, we show there is an oracle A such that $\mathrm{P}^A = \mathrm{UP}^A$ yet symmetry of information does not hold relative to A.

6.6.3. Theorem. *There is an oracle A such that $\mathrm{P}^A = \mathrm{UP}^A$ yet symmetry of information property (SMI) does not hold relative to A.*

Proof: An oracle A is constructed by Beigel, Buhrman, and Fortnow [BBF98] where

1. $\mathrm{P}^A = \oplus\mathrm{P}^A$, and
2. $\mathrm{NP}^A \neq \mathrm{EXP}^A$.

As $\mathrm{UP}^B \subseteq \oplus\mathrm{P}^B$ for any oracle B, item (1) implies that $\mathrm{P}^A = \mathrm{UP}^A$. By Valiant–Vazirani [VV86] we have that NP $\subseteq \mathrm{RP}^{\oplus\mathrm{P}}$, and thus the first item also implies $\mathrm{NP}^A = \mathrm{RP}^A$. As the proof of Theorem 6.6.2 relativizes, the assumption that symmetry of information holds implies $\mathrm{BPP}^A \neq \mathrm{EXP}^A$. This, however, contradicts the fact that $\mathrm{RP}^A = \mathrm{NP}^A$ and $\mathrm{NP}^A = \mathrm{EXP}^A$ from item (2). □

Chapter 7

Kolmogorov Complexity with Error

This chapter is based on the paper:

- L. Fortnow, T. Lee, and N. Vereshchagin. Kolmogorov complexity with error. To appear in Symposium on Theoretical Aspects of Computer Science, 2006.

7.1 Introduction

Thus far we have studied one modification of Kolmogorov complexity to bring it closer to real world data compression—namely by placing efficiency requirements on the decompression program which prints a string from its description.

Besides having little time, in the real world we also inevitably have to deal with errors. Suppose that we receive a string over a noisy or corrupted channel. Such a channel could change random bits of a string, possibly increasing its Kolmogorov complexity without adding any real information.

In some applications of data compression it is not essential that we maintain an exact copy of the data—in fact, sometimes it is preferable to trade some fidelity to the original data for the benefit of a more concise description. Such a compression scheme where some, less important we hope, information about the original data is lost is known as lossy compression. An important question in this setting is: what is the cheapest approximation of the data within our tolerance of distortion?

Intuitively, the scenarios of data corrupted over a noisy channel and lossy compression are in some sense complementary to one another: we expect that if we lossy compress a string received over a corrupted channel with our level of tolerance equal to the number of expected errors then the cheapest string within the level of tolerance will be the one with the high complexity noise removed. Ideally, that is, we would get back our original string. For certain compression schemes and models of noise this intuition can be made precise [Nat95].

In this chapter we explore a variation of Kolmogorov complexity designed to help us measure information in these settings. We define the Kolmogorov complexity of a string x with error a as the length of a smallest program generating a string x' that differs from x in at most a bits. We give tight bounds (up to logarithmic factors) on the maximum complexity of such strings and also look at time-bounded variations.

We also look at conditional Kolmogorov complexity with errors. Traditional conditional Kolmogorov complexity looks at the smallest program that converts a string y to a string x. In our context both x and y could be corrupted. We want the smallest program that converts a string close to y to a string close to x. We consider two variations of this definition, a uniform version where we have a single program that that converts any y' close to y to a string x' close to x and a nonuniform version where the program can depend on y'. We show examples giving a large separation between the uniform and nonuniform definitions.

Finally we consider symmetry of information for Kolmogorov complexity with error. Traditionally the complexity of the concatenation of strings x, y is roughly equal to the sum of the complexity of x and the complexity of y given x. We show that for any values of d and a the complexity of xy with error d is at most the sum of the complexity of x with error a and the complexity of converting a string y with $d-a$ error given x with a bits of error. We show the other direction fails in a strong sense, we do not get equality for any a.

7.2 Preliminaries

We use $d_H(x, y)$ to denote the Hamming distance between two binary strings x, y, that is the number of bits on which they differ. For $x \in \{0,1\}^n$ we let $B_n(x, R)$ denote the set of n-bit strings within Hamming distance R from x, and $V(n, R) = \sum_{i=1}^{R} \binom{n}{i}$ denote the volume of a Hamming ball of radius R over n-bit strings. For a real number $0 < \lambda \leq 1/2$ we write $H(\lambda) = -\lambda \log \lambda - (1-\lambda)\log(1-\lambda)$ for the binary entropy function. The binary entropy is useful in the following approximation of $V(n, R)$, which can be found in [CHLL97]:

7.2.1. Lemma. *Suppose that $0 < \lambda \leq 1/2$ and λn is an integer. Then*

$$\frac{2^{nH(\lambda)}}{\sqrt{8n\lambda(1-\lambda)}} \leq V(n, \lambda n) \leq 2^{nH(\lambda)}.$$

7.3 Definitions

We consider several possible ways of defining Kolmogorov complexity with error. In this section we present these alternatives in order to evaluate their relative merits in the coming sections.

7.3.1. Definition. Let $d : (\{0,1\}^n)^2 \to R$ be a metric, and $a \in R$. The complexity of x with error a, denoted $C_a(x)$ is $C_a(x) = \min_{x'}\{C(x') : d(x', x) \leq a\}$.

Although we give the definition for a general metric, in this chapter we will only consider the Hamming distance metric.

We will also consider a time bounded version of this definition, $C^t_a(x) = \min_{x'}\{C^t(x') : d(x, x') \leq a\}$, where $C^t(x \mid y)$ is the length of a shortest program p such that $U(p, y)$ prints x in less than t time steps.

A relative version of Kolmogorov complexity with error is defined by Impagliazzo, Shaltiel and Wigderson [ISW00]. That is, they use the definition $C_\delta(x) = \min\{C(y) : d_H(x, y) \leq \delta l(x)\}$. We prefer using absolute distance here as it behaves better with respect to concatenations of strings—using relative distance has the disadvantage of severe nonmonotonicity over prefixes. Take, for example, $x \in \{0,1\}^n$ satisfying $C(x) \geq n$. Let $y = 0^{2n}$. Then $C_{1/3}(x) \geq n - \log V(n, n/3)$ while $C_{1/3}(xy) \leq \log n + O(1)$. Using absolute error we have that $C_a(xy) \geq C_a(x) - O(\log n)$, that is it only suffers from logarithmic dips as with standard definition.

Defining conditional complexity with error is somewhat more subtle. We introduce both uniform and nonuniform versions of conditional complexity with error.

7.3.2. Definition. For a Turing machine T, the uniform conditional complexity, denoted $(C^u_{a,b})_T(x \mid y)$, is the length of a shortest program p such that, *for all* y' satisfying $d(y, y') \leq b$ it holds that $T(p, y')$ outputs a string whose distance from x is less than a.

The invariance theorem remains true: there is a universal machine U such that for any other Turing machine T, there is a constant c_T such that $(C^u_{a,b})_U(x \mid y) \leq (C^u_{a,b})_T(x \mid y) + c_T$, for all x, y, a, b. We fix such a U and drop the subscript.

7.3.3. Definition. Nonuniform conditional complexity, denoted $C_{a,b}(x \mid y)$ is defined as $C_{a,b}(x \mid y) = \max_{y'} \min_{x'}\{C(x' \mid y') : d(x', x) \leq a \text{ and } d(y', y) \leq b\}$.

In Section 7.6 we study the difference between these two measures.

7.4 Strings of Maximal Complexity

One of the most famous applications of Kolmogorov complexity is the incompressibility method (see [LV97], Chapter 6). To prove there exists an object with a certain property, we consider an object with maximal Kolmogorov complexity and show that it could be compressed if it did not possess this property. We have seen an example of this type of proof in Chapter 1 to prove lower bound on Ramsey numbers.

This method relies on the first pillar of Kolmogorov complexity: for every length n, there is a string x of complexity at least n. It is also easy to see that, up to an additive constant, every string has complexity at most its length. What is the behavior of maximal complexity strings in the error case? For this discussion, we restrict ourselves to the Hamming distance case.

Again by a counting argument, we see that for every n there is an x of length n with $C_a(x) \geq \log 2^n/V(n,a) = n - \log V(n,a)$. Upper bounding the complexity of strings in the error case requires a bit more work, and has a close connection with the construction of covering codes. A covering code $\mathcal{C}$ of radius a is a set of strings such that for every $x \in \{0,1\}^n$ there is an element $y \in \mathcal{C}$ such that $d_H(x,y) \leq a$. Thus an upper bound on the maximum complexity strings will be given by the existence of covering codes of small size. The following Lemma is well known in the covering code literature, (see [CHLL97] or [KSV03]).

7.4.1. Lemma. *For any n and integer $R \leq n$, there exists a set $\mathcal{C} \subseteq \{0,1\}^n$ with the following properties:*

1. $|\mathcal{C}| \leq n2^n/V(n,R)$
2. *for every $x \in \{0,1\}^n$, there exists $c \in \mathcal{C}$ with $d_H(x,c) \leq R$*
3. *The set $\mathcal{C}$ can be computed in time* $\mathrm{poly}(2^n)$

Proof: For the first two items we argue by the probabilistic method. The third item will be obtained by derandomizing this argument with the method of conditional probabilities.

Fix a point $x \in \{0,1\}^n$. We uniformly at random choose k elements $x_1, \ldots, x_k$ of $\{0,1\}^n$. The probability P_x that x is not contained in $\cup_{i=1}^k B(x_i,R)$ is precisely

$$P_x = (1 - V(n,R)/2^n)^k \leq e^{-kV(n,R)/2^n}$$

For the inequality we have used the fact that $e^z \geq 1 + z$ for any z. Taking k to be $n2^n/V(n,R)$ makes this probability strictly less than 2^{-n}. Thus the probability of the union of the events P_x over $x \in \{0,1\}^n$ is, by the union bound, less than 1 and there exists a set of $n2^n/V(n,R)$ centers which cover $\{0,1\}^n$. This gives items 1 and 2.

For item 3 we now derandomize this argument using the method of conditional probabilities. Let $t = n2^n/V(n,R)$ be the desired size of our covering. We now define successively $x_1, \ldots, x_t$ so that the following invariant is true for all $k \leq t$:

$$\sum_x \Pr[x \text{ is not contained in } \cup_{i=1}^k B(x_i,R) \cup \cup_{i=1}^{t-k} B(y_i,R)] < 1. \tag{7.1}$$

The probability is taken over randomly chosen $y_1, \ldots, y_{t-i}$ and the sum is over all strings x of length n.

For $k = 0$ the invariant is true as shown above. Assume now that $x_1, \dots, x_{k-1}$ are already defined so that Equation (7.1) is true for $k-1$. We claim that there is x_k such that Equation (7.1) is true for k. Note that for all x the probability $p(x)$ that x is not contained in $\cup_{i=1}^{k-1} B(x_i, R) \cup \cup_{i=1}^{t-k+1} B(y_i, R)$ is the average over x_k of the the probability that x is not contained in $\cup_{i=1}^{k} B(x_i, R) \cup \cup_{i=1}^{t-k} B(y_i, R)$. This applies for the sum in Equation (7.1) as well hence there is x_k satisfying the invariant.

For every given x_k the value of the left hand side of Equation (7.1) can be calculated in time poly(2^n), since the probability $p(x)$ that x is not contained in a set $A \cup \cup_{i=1}^{t-k} B(y_i, R)$ is equal to 0 if $x \notin A$ and to $(1 - V(n, R)/2^n)^{t-k}$ otherwise. The number of different x_k is 2^n. Therefore an appropriate x_k can be found in time poly(2^n).

As the probability $p(x)$ depends only on whether x is in $\cup_{i=1}^{k-1} B(x_i, R)$ or not, we just choose the string x_k that maximizes the cardinality of the set $\cup_{i=1}^{k} B(x_i, R)$. So our algorithm is a greedy one: each time it chooses a ball that covers maximal number of non-covered elements. □

Actually, there is a general theorem on the quality of the greedy algorithm to choose subcoverings.

7.4.2. Theorem (Corollary 37.5 in [CLR90]). *For any set X and covering $S_1, \dots, S_N$ of X, the greedy algorithm chooses a sub-covering of $S_1, \dots, S_N$ that has at most* $\ln|X| + 1$ *times more sets than the minimal sub-covering.*

From this theorem we can derive item 3 of Lemma 7.4.1 with a slightly weaker bound $|\mathcal{C}| \le O(n^2 2^n / V(n, R))$.

7.4.3. Theorem. *For every n, a and $x \in \{0,1\}^n$, $C_a(x) \le n - \log V(n, a) + O(\log n)$.*

Proof: By Lemma 7.4.1 we know that a covering code with radius a of cardinality less than $n 2^n / V(n, a)$ exists. Let $\mathcal{C}$ be the lexicographically first such covering. Such a covering can be described by saying "look for the lexicographically first covering over $\{0,1\}^n$ of radius a", and thus has a description of size $O(\log n)$. For any $x \in \{0,1\}^n$ there is an element $c \in \mathcal{C}$ such that $d_H(x, c) \le a$. Once we know the covering, this element c can be described by its index in the covering, of size $n + \log n - \log V(n, a)$. Thus the total description is of size $n - \log V(n, a) + O(\log n)$. □

One nice property of covering codes is that they behave very well under concatenation. Let $\mathcal{C}_1$ be a covering code of $\{0,1\}^{n_1}$ of radius R_1 and $\mathcal{C}_2$ be a covering code of $\{0,1\}^{n_2}$ of radius R_2. Now let $\mathcal{C} = \{cc' : c \in \mathcal{C}_1, c' \in \mathcal{C}_2\}$ be the set of all ordered concatenations of codewords from $\mathcal{C}_1$ with codewords from $\mathcal{C}_2$. Then $\mathcal{C}$ is a covering code over $\{0,1\}^{n_1+n_2}$ of radius $R_1 + R_2$.

We can use this idea in combination with item 3 of Lemma 7.4.1 to efficiently construct near-optimal covering codes. This construction has already been used for a complexity-theoretic application in [DGHS00].

7.4.4. THEOREM. *There is a polynomial time bound $p(n)$ such that $C_d^{p(n)}(x) \leq n - \log V(n,d) + O(n \log\log n/\log n)$ for every $x \in \{0,1\}^n$ and every d.*

Proof: We construct a covering code over $\{0,1\}^n$ with radius d such that the ith element of the covering can be generated in time polynomial in n. Let $\ell = \log n$ and divide n into n/ℓ blocks of length ℓ. Let $r = (d/n)\ell$. Now by item 3 of Lemma 7.4.1 we can in time polynomial in n construct a covering code over $\{0,1\}^\ell$ of radius r and of cardinality $\ell 2^\ell / V(\ell, r)$. Call this covering $\mathcal{C}_\ell$. Our covering code $\mathcal{C}$ over $\{0,1\}^n$ will be the set of codewords $\{c_1 c_2 \cdots c_{n/\ell} : c_i \in \mathcal{C}_\ell\}$. The size of this code will be:

$$\begin{aligned} |\mathcal{C}| &\leq (2^{\ell - \log V(\ell,r) + \log \ell})^{n/\ell} = (2^{\ell - \ell H(d/n) + O(\log \ell)})^{n/\ell} \\ &= 2^{n - nH(d/n) + O(n \log \ell/\ell)} = 2^{n - \log V(n,d) + O(n \log \ell/\ell)}. \end{aligned} \tag{7.2}$$

The second and the last inequalities hold by Lemma 7.2.1.

In this proof we assumed that all numbers $\log n$, $n/\log n$, and $d \log n/n$ are integer. In the general case we can let $\ell = \lfloor \log n \rfloor$, $m = \lceil n/\ell \rceil$ and $r = \lfloor d/m \rfloor$. We divide n into m blocks of length ℓ or $\ell - 1$. We will need two covering codes: of lengths ℓ and $\ell - 1$, both of radius r. Both codes can be found in time polynomial in n. It is easy to verify that the inequalities Equation (7.2) remain true up to an error term $O(n \log \ell/\ell)$ in the exponent. □

7.5 Dependence of Complexity on the Number of Allowed Errors

Both the uniform and the non-uniform conditional complexities $C^u_{a,b}$ and $C_{a,b}$ are decreasing functions in a and increasing in b. Indeed, if b decreases and a increases then the number of y''s decreases and the number of x''s increases, thus the problem to transform every y' to some x' becomes easier. What is the maximal possible rate of this decrease/increase? For the uniform complexity, we have no non-trivial bounds. For the non-uniform complexity, we have the following

7.5.1. THEOREM. *For all x, y of length n and all $a \leq a'$, $b' \leq b$ it holds*

$$C_{a,b}(x \mid y) \leq C_{a',b'}(x \mid y) + \log(V(n,a)/V(n,a')) + \log(V(n,b')/V(n,b)) + O(\log n).$$

Proof: Let y' be a string at distance b from y. We need to find a short program mapping it to a string at distance a from x. To this end we need the following lemma from [VV04a].

7.5.2. Lemma. *For all $d \le d' \le n$ having the form i/n every Hamming ball of radius d' in the set of binary strings of length n can be covered by at most $O(n^4 V(n,d')/V(n,d))$ Hamming balls of radius d.*

Apply the lemma to $d' = b$, $d = b'$ and to the ball of radius b centered at y'. Let $B_1, \dots, B_N$, where $N = O(n^4 V(n,b)/V(n,b'))$, be the covering balls. Let B_i be a ball containing the string y and let y'' be its center. There is a program, call it p, of length at most $C_{a',b'}(x \,|\, y)$ mapping y'' to a string at distance a' from x. Again apply the lemma to $d = a$, $d' = a'$ and to the ball of radius d' centered at x'. Let $C_1, \dots, C_M$, where $M = O(n^4 V(n,a')/V(n,a))$, be the covering balls. Let C_j be a ball containing the string x and let x'' be its center. Thus x'' is at distance a from x and can be found from y', p, i, j. This implies that $K(x'' \,|\, y') \le l(p) + \log N + \log M + O(\log n)$ (extra $O(\log n)$ bits are needed to separate p, i and j). □

In the above proof, it is essential that we allow the program mapping y' to a string close to x depend on y'. Indeed, the program is basically the triple (p, i, j) where both i and j depend on y'. Thus the proof is not valid for the uniform conditional complexity. And we do not know whether the statement itself is true for the uniform complexity.

A similar inequality holds for time bounded complexity.

7.5.3. Theorem. *There is a polynomial p such that for all x, y of length n and all $a \le a'$, $b' \le b$, and all t it holds*

$$\begin{aligned} C^{p(n,t)}_{a,b}(x \,|\, y) \le C^t_{a',b'}(x \,|\, y) + \log(V(n,a)/V(n,a')) &+ \log(V(n,b')/V(n,b)) \\ &+ O(n \log\log n / \log n). \end{aligned}$$

Proof: The proof is entirely similar. We just need the following time bounded version of Lemma 7.5.2:

7.5.4. Lemma. *For all $d \le d' \le n$ having the form i/n every Hamming ball of radius d' in the set of binary strings of length n can be covered in time $O(2^n)$ by at most $O(n^5 V(n,d')/V(n,d))$ Hamming balls of radius d.*

This lemma follows immediately from Lemma 7.5.2 and Theorem 7.4.2. Using the method of conditional probabilities we can prove a slightly better bound $O(n^4 V(n,d')/V(n,d))$ but this needs going into details of the proof of Lemma 7.5.2. □

7.6 Uniform vs. Nonuniform Conditional Complexity

In this section we show an example where the uniform version of conditional complexity can be exponentially larger than the nonuniform one. Our example will be for $C_{0,b}(x \mid x)$. Notice that this example is the error correction problem. Given some x' such that $d_H(x, x') \leq b$, we want to recover x exactly. The intuition behind the proof is the following: say we have some computable family S of Hamming balls of radius b, and let x be the center of one of these balls. Given any x' such that $d_H(x, x') \leq b$, there may be other centers of the family S which are also less than distance b from x'. Say there are k of them. Then x has a nonuniform description of size about $\log k$ by giving the index of x in the k balls which are of distance less than b from x'.

In the uniform case, on the other hand, our program can no longer be tailored for a particular x', it must work for any x' such that $d_H(x, x') \leq b$. That is, intuitively, the program must be able to distinguish the ball of x from any other ball intersecting the ball of x. To create a large difference between the nonuniform and uniform conditional complexity measures, therefore, we wish to construct a large family of Hamming balls, every two of which intersect, yet that no single point is contained in the intersection of too many balls. Moreover, we want $C_{0,b}(x \mid x)$ be much larger than even $C^u_{a,b}(x \mid x)$ for a non-negligible a. Thus we want the contractions of every two balls to radius a be disjoint. The next lemma shows the existence of such a family.

7.6.1. Lemma. *For every length m of strings and a, b, and N satisfying the inequalities*

$$N^2 V(m, 2a) \leq 2^{m-1}, \quad N^2 V(m, m-2b) \leq 2^{m-1}, \quad NV(m, b) \geq m2^{m+1} \qquad (7.3)$$

there are strings $x_1, \ldots, x_N$ such that the balls of radius a centered at $x_1, \ldots, x_N$ are pairwise disjoint, and the balls of radius b centered at $x_1, \ldots, x_N$ are pairwise intersecting but no string belongs to more than $NV(m, b)2^{1-m}$ of them.

Proof: The proof is by probabilistic arguments. Take N independent random strings $x_1, \ldots, x_N$. We will prove that with high probability they satisfy the statement.

First we estimate the probability that there are two intersecting balls of radius a. The probability that two fixed balls intersect is equal to $V(m, 2a)/2^m$. The number of pairs of balls is less than $N^2/2$, and by a union bound, there are two intersecting balls of radius a with probability at most $N^2 V(m, 2a)/2^{m+1} \leq 1/4$ (use the first inequality in Equation (7.3)).

Let us estimate now the probability that there are two disjoint balls of radius b. If the balls of radius b centered at x_j and x_i are disjoint then x_j is at distance at most $m - 2b$ from the string $\bar{x}_i$, that is obtained from x_i by flipping all bits. Therefore the probability that for a fixed pair (i, j) the balls are disjoint is at most $V(m, m-2b)/2^m$. By the second inequality in Equation (7.3), there are two disjoint balls with probability at most 1/4.

It remains to estimate the probability that there is a string that belongs to more than $NV(m, b)2^{1-m}$ balls of radius b. Fix x. For every i the probability that x lands in B_i, the ball of radius b centered at x_i, is equal to $p = |B_i|/2^m = V(m, b)/2^m$. So the average number of i with $x \in B_i$ is $pN = NV(m, b)/2^m$. By Chernoff inequality the probability that the number of i such that x lands in B_i exceeds twice the average is at most

$$\exp(-pN/2) = \exp(-NV(m, b)/2^{m+1}) \leq \exp(-m) \ll 2^{-m}$$

(use the third inequality in Equation (7.3)). Thus even after multiplying it by 2^m the number of different x's we get a number close to 0. □

Using this lemma we find x with exponential gap between $C_{0,b}(x \,|\, x)$ and $C^u_{0,b}(x \,|\, x)$ and even between $C_{0,b}(x \,|\, x)$ and $C^u_{a,b}(x \,|\, x)$ for a, b linear in the length n of x. Namely fix some rational constants α, β, γ satisfying

$$0 < \alpha < 1/4 < \beta < 1/2, \quad 2H(\beta) > 1+H(2\alpha), \quad 2H(\beta) > 1+H(1-2\beta), \quad \gamma \geq 1. \tag{7.4}$$

Note that if β is close to 1/2 and α is close to 0 then the inequalities Equation (7.4) are fulfilled.

7.6.2. Theorem. *For all sufficiently large m there is a string x of length $n = \gamma m$ with $C_{0,\beta m}(x \,|\, x) = O(\log m)$ while $C^u_{\alpha m,\beta m}(x \,|\, x) \geq m(1 - H(\beta)) - O(\log m)$.*

Proof: Given m let $a = \alpha m$, $b = \beta m$ and $N = m2^{m+1}/V(m, b)$. Let us verify that for large enough m the inequalities Equation (7.3) in the condition of Lemma 7.6.1 are fulfilled. Taking the logarithm of the first inequality Equation (7.3) and ignoring all terms of order $O(\log m)$ we obtain

$$2(m - mH(\beta)) + mH(2\alpha) < m$$

This is true by the second inequality in Equation (7.4). Here we used that $\log V(m, b) = mH(\beta)$ and $\log V(m, 2a) = mH(2\alpha)$ (ignoring logarithmic terms), as both $\beta, 2\alpha$ are less than 1/2. Taking the logarithm of the second inequality Equation (7.3) we obtain

$$2(m - mH(\beta)) + mH(1 - 2\beta) < m.$$

This is implied by the third inequality in Equation (7.4). Finally, the last inequality Equation (7.3) holds by the choice of N.

Find the first sequence $x_1, \ldots, x_N$ satisfying the lemma. This sequence has complexity at most $C(m) = O(\log m)$. Append 0^{n-m} to all strings $x_1, \ldots, x_N$. Obviously the resulting sequence also satisfies the lemma. For each string x_i we have $C_{0,b}(x_i \,|\, x_i) = O(\log m)$, as given any x' at distance at most b from x_i we can specify x_i by specifying its index among centers of the balls in the family containing x' in $\log(NV(m,b)2^{1-m}) = \log 4m$ bits and specifying the family itself in $O(\log m)$ bits.

It remains to show that there is x_i with $C^u_{a,b}(x_i \,|\, x_i) \geq \log N$. Assume the contrary and choose for every x_i a program p_i of length less than $\log N$ such that $U(p, x')$ is at distance a from x_i for every x' at distance at most b from x_i. As N is strictly greater than the number of strings of length less than $\log N$, by the Pigeon Hole Principle there are different x_i, x_j with $p_i = p_j$. However the balls of radius b with centers x_i, x_j intersect and there is x' at distance at most b both from x_i, x_j. Hence $U(p, x')$ is at distance at most a both from x_i, x_j, a contradiction. □

Again, at the expense of replacing $O(\log m)$ by $O(m \log\log m / \log m)$ we can prove an analog of Theorem 7.6.2 for time bounded complexity.

7.6.3. Theorem. *There is a polynomial p such that for all sufficiently large m there is a string x of length $n = \gamma m$ with $C^{p(n)}_{0,\beta m}(x \,|\, x) = O(m \log\log m / \log m)$ while $C^u_{\alpha m, \beta m}(x \,|\, x) \geq m(1 - H(\beta)) - O(m \log\log m / \log m)$. (Note that C^u has no time bound; this makes the statement stronger.)*

Proof: We use a similar code concatenation argument as in Theorem 7.4.4. First we need to derandomize Lemma 7.6.1. We claim that a list of strings $x_1, \ldots, x_N$ satisfying Lemma 7.6.1 can be found in time $\mathrm{poly}(2^n)$. This again can be done using the method of conditional probabilities. We define successively $x_1, \ldots, x_k$ so that the following invariant is true: the sum $S_k = A + B + C$ is less than 1. Here: A is the sum over all $i < j \leq N$ of the probability that the balls of radius a centered at x_i and x_j intersect; B is the sum over all $i < j \leq N$ of the probability that the balls of radius b centered at x_i and x_j are disjoint; C is the sum over all x of the probability that the string x belongs to more than $NV(m,b)2^{1-m}$ balls of radius b centered at $x_1, \ldots, x_N$. The strings $x_{k+1}, \ldots, x_N$ are chosen at random.

To make the induction step, given $x_1, \ldots, x_{k-1}$ such that $S_{k-1} < 1$ we need to find x_k satisfying $S_k < 1$. The sum S_{k-1} can be considered as the average over all choices of x_k of the sum S_k. Therefore there is x_k such that $S_k < 1$. We just look through all x_k until we find x_k with $S_k < 1$. The key observation allowing to do this in time $\mathrm{poly}(2^n)$ is that all three sums A, B, C are rational numbers with denominator 2^{mN} that can be calculated exactly in time $\mathrm{poly}(2^n)$ given $x_1, \ldots, x_k$. This is easy for A and B: for fixed i, j the corresponding probability in the sum A is equal either to $0/1$ (when $i, j \leq k$) or to $V(m,a)/2^m$. The same argument applies to the sum B.

The sum C. Fix x and let l be the number of balls with indices among $1, \dots, k$ the string x belongs to. We need to calculate the probability that x belongs to more than $R = NV(m, b)2^{1-m} - l$ balls of radius b centered at $x_1, \dots, x_N$. The probability that among $N-k$ independent trials with probability $p = V(m, b)2^{-m}$ of success there are at least R successful trials is equal to the sum

$$\sum_{i=R}^{N-k} \binom{N-k}{i} p^i (1-p)^{N-k-i}.$$

As p is a rational with denominator 2^m, this probability is a rational with denominator $2^{m(N-k)}$ and can be calculated exactly in time $\mathrm{poly}(N-k, m) = \mathrm{poly}(2^n)$.

To finish the proof we need to verify that the statement in the conclusion of Lemma 7.6.1 behaves well under the concatenation of codes. Apply (the derandomized version of) Lemma 7.6.1 to $m' = \log m$, $a' = a/k$, $b' = b/k$ where $k = m/\log m$ and $N' = m'2^{m'+1}/V(m', b')$. Let $\mathcal{C} = \{x_1, \dots, x_{N'}\}$ be the resulting code, which can be found in time $\mathrm{poly}(2^{m'}) = \mathrm{poly}(m)$. Let $\mathcal{C}^k = \{x_1, \dots, x_N\}$ be the product code, where $N = (N')^k = (m'2^{m'+1}/V(m', b')))^k$. Then the balls of radius a centered at strings $x_1, \dots, x_N$ are pairwise disjoint, the balls of radius b centered at strings $x_1, \dots, x_N$ are pairwise intersecting, and no string belongs to more than $(N'V(m', b')2^{1-m'})^k = (4m')^k$ of them. The latter implies that

$$C_{0,b}^{\mathrm{poly}(m)}(x \,|\, x) \le k\log(4m') + O(\log m) = O(m \log\log m/\log m).$$

To lower bound $C^u_{\alpha m, \beta m}(x \,|\, x)$ we need to lower bound N. We have

$$\begin{aligned} \log N = k \log N' &= k(m' - m'H(\beta) + O(\log m')) \\ &= m - mH(\beta) + O(m \log\log m/\log m), \end{aligned}$$

which implies the lower bound

$$C^u_{\alpha m, \beta m}(x \,|\, x) \ge m(1 - H(\beta)) - O(m \log\log m/\log m).$$

□

7.7 Symmetry of Information

In this section, we investigate how the fourth pillar of Kolmogorov complexity, symmetry of information, stands in the setting with errors.

We see that the easy direction of symmetry of information is again easy: The inequality $C_d(xy) \le C_a(x) + C_{d-a,a}(y \,|\, x) + O(\log n)$ holds for any a — let p be a program of length $C_a(x)$ which prints a string x^* within Hamming distance a of x. Let q be a shortest program which, given x^*, prints a string y^* within Hamming distance $d-a$ of y. By definition, $C_{d-a,a}(y \,|\, x) = \max_{x'} \min_{y'} C(y' \,|\, x') \ge$

$\min_{y'} C(y' \mid x^*) = l(q)$. Now given p and q and a way to tell them apart, we can print the string xy within d errors.

For the converse direction we would like to have the statement

$$\begin{array}{l}\text{For every } d, x, y \text{ there exists } a \text{ such that}\\ C_d(xy) \geq C_a(x) + C_{d-a,a}(y \mid x) - O(\log n).\end{array} \tag{7.5}$$

We do not expect this statement to hold for every a, as the shortest program for xy will have a particular pattern of errors which might have to be respected in the programs for x and y given x. We now show, however, that even the Property 7.5 is too much to ask.

7.7.1. Theorem. *For every n and all $d \leq n/4$ there exist $x, y \in \{0,1\}^n$ such that for all $a \leq d$ the difference*

$$\Delta(a) = (C_a(y) + C_{d-a,a}(x \mid y)) - C_d(xy)$$

is more than both

$$\log V(n,d) - \log V(n,a), \qquad \log V(n,d+a) - \log V(n,d-a) - \log V(n,a),$$

up to an additive error term of the order $O(\log n)$.

Before proving the theorem let us show that in the case, say, $d = n/4$ it implies that for some positive ϵ we have $\Delta(a) \geq \epsilon n$ for all a. Let $\alpha < 1/4$ be the solution to the equation

$$H(1/4) = H(1/4+\alpha) - H(1/4-\alpha).$$

Note that the function in the right hand side increases from 0 to 1 as α increases from 0 to 1/4. Thus this equation has the unique solution.

7.7.2. Corollary. *Let $d = n/4$ and let x, y be the strings existing by Theorem 7.7.1. Then we have $\Delta(a) \geq n(H(1/4) - H(\alpha)) - O(\log n)$ for all a.*

Proof: Let $z = a/n$. Up to an additive error term of the order $O(\log n)$ the first lower bound for $\Delta(a)$ in Theorem 7.7.1 is equal to $n(H(1/4) - H(z))$ and the second one to $n(H(1/4+z) - H(1/4-z) - H(z))$. Thus we need to prove that for all $z \leq 1/4$ at least on of the two functions

$$H(1/4) - H(z), \quad H(1/4+z) - H(1/4-z) - H(z)$$

exceeds $H(1/4) - H(\alpha)$. Note that α is chosen so that in the point $z = \alpha$ these two functions coincide. Therefore it is enough to show that on the interval $[0, \alpha]$ the first function decreases, and on the segment $[\alpha, 1/4]$ the second function increases. The former assertion is easy: the function $H(1/4) - H(z)$ is decreasing z on the segment $[0, 1/2]$, which includes $[\alpha, 1/4]$.

Let us prove that the function $H(1/4+z) - H(1/4-z) - H(z)$ increases on the interval $[\alpha, 1/4]$. As the derivative of $H(z)$ decreases on $[0, 1/2]$, the function $-H(1/4-z) - H(z)$ (and hence $H(1/4+z) - H(1/4-z) - H(z)$) increases on the interval $[1/8, 1/4]$. Thus it suffices to prove that $1/8 \le \alpha$, that is, $H(1/4) + H(1/4-1/8) < H(1/4+1/8)$. This is implied by the general inequality $H(a+b) + H(a+c) \le H(a+b+c) + H(a)$ (let $a = 0$, $b = 1/4$, $c = 1/8$). This inequality holds for every function with non-positive second derivative, for all a and all non-negative b, c. Indeed, let say $b \ge c$. Then by Lagrange's theorem we have

$$H(a+b+c) - H(a+b) = H'(\theta) \cdot c \le H'(\eta) \cdot c = H(a+c) - H(a),$$

where θ is a point in the segment $[a+b, a+b+c]$ and η is a point in the segment $[a, a+c]$. The middle inequality is true, as the interval $[a+b, a+b+c]$ is to the right of the interval $[a; a+c]$. The case $c \ge b$ is entirely similar. □

Proof:[of Theorem 7.7.1] Coverings will again play an important role in the proof. Let $\mathcal{C}$ be the lexicographically first minimal size covering of radius d. Choose y of length n with $C(y) \ge n$, and let x be the lexicographically least element of the covering within distance d of y. Notice that $C_d(xy) \le n - \log V(n,d)$, as the string xx is within distance d of xy, and can be described by giving a shortest program for x and a constant many more bits saying "repeat." (In the whole proof we neglect additive terms of order $O(\log n)$). Let us prove first that $C(x) = n - \log V(n,d)$ and $C(y \mid x) = \log V(n, d_1) = \log V(n,d)$, where d_1 stands for the Hamming distance between x and y. Indeed,

$$\begin{aligned} n \le C(y) \le C(x) + C(y \mid x) &\le n - \log V(n,d) + C(y \mid x) \\ &\le n - \log V(n,d) + \log V(n, d_1) \le n. \end{aligned}$$

Thus all inequalities here are equalities, hence $C(x) = n - \log V(n,d)$ and $C(y \mid x) = \log V(n, d_1) = \log V(n,d)$.

Let us prove now the first lower bound for $\Delta(a)$. As y has maximal complexity, for any $0 \le a \le d$ we have $C_a(y) \ge n - \log V(n,a)$. Summing the inequalities

$$\begin{aligned} -C_d(xy) &\ge -n + \log V(n,d), \\ C_a(y) &\ge n - \log V(n,a), \\ C_{d-a,a}(x \mid y) &\ge 0, \end{aligned}$$

we obtain the lower bound $\Delta(a) \ge \log V(n,d) - \log V(n,a)$. To prove the second lower bound of the theorem, we need to show that

$$C_{d-a,a}(x \mid y) \ge \log V(n, d+a) - \log V(n, d-a) - \log V(n,d). \qquad (7.6)$$

To prove that $C_{d-a,a}(x \mid y)$ exceeds a certain value v we need to find a y' at distance at most a from y such that $C(x' \mid y') \ge v$ for all x' at distance at most

$d-a$ from x. Let y' be obtained from y by changing a random set of a bits on which x and y agree. This means that $C(y' \mid y, x) \geq \log V(n-d_1, a)$. It suffices to show that

$$C(x \mid y') \geq \log V(n, d+a) - \log V(n,d).$$

Indeed, then for all x' at distance at most $d-a$ from x we will have

$$C(x' \mid y') + \log V(n, d-a) \geq C(x \mid y')$$

(knowing x' we can specify x by its index in the ball of radius $d-a$ centered at x'). Summing these inequalities will yield Equation (7.6).

We use symmetry of information in the non-error case to turn the task of lower bounding $C(x \mid y')$ into the task of lower bounding $C(y' \mid x)$ and $C(x)$. This works as follows: by symmetry of information,

$$C(xy') = C(x) + C(y' \mid x) = C(y') + C(x \mid y').$$

As $C(y')$ is at most n, using the second part of the equality we have $C(x \mid y') \geq C(x) + C(y' \mid x) - n$. Recall that $C(x) = n - \log V(n,d)$. Thus to complete the proof we need to show the inequality $C(y' \mid x) \geq \log V(n, d+a)$, that is, y' is a random point in the Hamming ball of radius $d+a$ with the center at x. To this end we first note that $\log V(n, d+a) = \log V(n, d_1+a)$ (up to a $O(\log n)$ error term). Indeed, as $a+d \leq n/2$ we have $\log V(n, d+a) = \log \binom{n}{d+a}$ and $\log V(n,d) = \log \binom{n}{d}$. The same holds with d_1 in place of d. Now we will show that $\log V(n,d) - \log V(n,d_1) = O(\log n)$ implies that $\log V(n, d+a) - \log V(n, d_1+a) = O(\log n)$. It is easy to see that $\binom{n}{d+1}/\binom{n}{d_1+1} \leq \binom{n}{d}/\binom{n}{d_1}$ provided $d_1 \leq d$. Using the induction we obtain $\binom{n}{d+a}/\binom{n}{d_1+a} \leq \binom{n}{d}/\binom{n}{d_1}$.

Thus we have

$$\log V(n, d+a) - \log V(n, d_1+a) = \log\left(\binom{n}{d+a}\Big/\binom{n}{d_1+a}\right)$$
$$\leq \log\left(\binom{n}{d}\Big/\binom{n}{d_1}\right) = \log V(n,d) - \log V(n,d_1) = O(\log n).$$

Again we use (the conditional form of) symmetry of information:

$$C(y'y \mid x) = C(y \mid x) + C(y' \mid y, x) = C(y' \mid x) + C(y \mid y', x).$$

The string y differs from y' on a bits out of the $d_1 + a$ bits on which y' and x differ. Thus $C(y \mid y', x) \leq \log \binom{d_1+a}{a}$. Now using the second part of the equality we have

$$\begin{aligned} C(y' \mid x) &= C(y \mid x) + C(y' \mid y, x) - C(y \mid y', x) \\ &\geq \log V(n, d_1) + \log V(n-d_1, a) - \binom{d_1+a}{a}. \end{aligned}$$

We have used that $\log V(n-d_1,a) = \log\binom{n-d_1}{a}$, as $a \leq (n-d_1)/2$. Hence,

$$\begin{aligned} C(y' \mid x) &\geq \log\binom{n}{d_1} + \log\binom{n-d_1}{a} - \log\binom{d_1+a}{a} \\ &= \log\binom{n}{d_1+a} = \log V(n,d+a). \end{aligned}$$

□

Again, at the expense of replacing $O(\log n)$ by $O(n \log\log n/\log n)$ we can prove an analog of Theorem 7.7.1 for time bounded complexity:

7.7.3. Theorem. *There is a polynomial p such that for every n and all $d \leq n/4$ there exist $x, y \in \{0,1\}^n$ such that for all $a \leq d$ the difference*

$$\Delta(a) = (C_a(y) + C_{d-a,a}(x \mid y)) - C_d^{p(n)}(xy)$$

is larger than both values

$$\log V(n,d) - \log V(n,a), \qquad \log V(n,d+a) - \log V(n,d-a) - \log V(n,a),$$

up to an additive error term of the order $O(n \log\log n/\log n)$. Note that there is no time bound in $C_a(y) + C_{d-a,a}(x \mid y)$. This makes the statement stronger.

Proof: Use the code $\mathcal{C}$ from the proof to Theorem 7.4.4. □

Part II: Formula Size Lower Bounds

Chapter 8

Formula Size Lower Bounds

8.1 Introduction

In this part of the thesis we turn to the problem of proving lower bounds. The task of proving bounds is quite daunting—to show that a problem can be solved within a certain resource bound we just have to exhibit a single algorithm which works; to show a lower bound on the resources to solve a problem, we have to show that *any conceivable* algorithm using those resources cannot do the trick. For this reason, proving lower bounds is a notoriously difficult task. Many turning points in mathematics were the result of proving lower bounds:

- Around 1843 Galois showed that in general the roots of a fifth degree polynomial are not expressible in terms of the coefficients of the polynomial, the arithmetic operations addition, subtraction, multiplication, and division, and the application of radicals. This remarkable result also gave lower bounds on that complexity class of antiquity, constructions with a straightedge and compass.

- In 1851, Liouville showed that there is a real number which is not expressible as the root of a polynomial equation with integer coefficients. Perhaps more importantly for complexity, in 1874 Cantor showed that almost all real numbers are not so expressible. The technique he used, diagonalization, is still one of the best ways to prove lower bounds in complexity theory.

- In 1931 Gödel showed his famous Incompleteness Theorem—there are statements which can be formulated in arithmetic but can neither be proved nor disproved within the theory.

The current frontier of lower bounds is the P versus NP question. Gödel was again one of the first to realize the significance of this problem. In a letter to von Neumann in March of 1956, he says:

> “Man kann offenbar leicht eine Turingmaschine konstruieren, welche von jeder Formel F des engeren Funktionenkalküls u. jeder natürl. Zahl n zu entscheiden gestattet, ob F einen Beweis der Länge n hat (Länge = Anzahl der Symbole). Sei $\psi(F, n)$ die Anzahl der Schritte, die die Maschine dazu benötigt u. sei $\phi(n) = \max_F \psi(F, n)$. Die Frage ist, wie rasch $\phi(n)$ für eine optimale Maschine wächst. Man kann zeigen $\phi(n) \geq k \cdot n$. Wenn es wirklich eine Maschine mit $\phi(n) \sim k \cdot n$ (oder auch nur $\sim k \cdot n^2$) gäbe, hätte das Folgerungen von der grössten Tragweite.”

> [“One can obviously easily construct a Turing machine, which for every formula F in first order predicate logic and every natural number n, allows one to decide if there is a proof of F of length n (length = number of symbols). Let $\psi(F, n)$ be the number of steps the machine requires for this and let $\phi(n) = \max_F \psi(F, n)$. The question is how fast $\phi(n)$ grows for an optimal machine. One can show that $\phi(n) \geq k \cdot n$. If there really were a machine with $\phi(n) \sim k \cdot n$ (or even $\sim k \cdot n^2$), this would have consequences of the greatest importance.”]

Although later in the same letter Gödel says that it is “within the realm of possibility” that $\phi(n)$ grows like n^2, which would in particular imply P = NP, the current consensus among researchers is that P $\neq$ NP.

One approach to proving P $\neq$ NP has focused on proving lower bounds for circuits. It is still a major open problem to even prove a superlinear circuit size lower bound for a function in NP—the current best bound is $5n - o(n)$ [LR01, IM02]. As proving lower bounds for circuits still seems excruciatingly difficult, we will focus in this part of the thesis on the weaker model of formula size. A formula is a binary tree with internal nodes labelled by AND and OR gates, and leaves labelled by literals, that is a variable or its negation. A crucial difference between a formula and a circuit is that in a formula every gate has fan-out exactly one, whereas a circuit can have larger fan-out. This restriction is the key property which allows us to prove larger lower bounds for formulas than for circuits.

A good example to illustrate the difference between circuits and formulas is the PARITY function which evaluates to 1 on input a binary string x if and only if the number of ones in x is odd. Assuming for simplicity that the input length is $n = 2^\ell$, a natural construction for PARITY which comes to mind is a binary tree with XOR gates labelling all the internal nodes, where an XOR gate computes the parity of 2 bits. When we expand a single XOR gate into a formula with AND and OR gates and negations, we find that in order to keep fan-out 1 we end up with a formula of size 4. This blow-up continues as we expand the XOR gates in the binary tree so that at the end we have a formula of size $4^\ell = n^2$.

Khrapchenko was the first to show that this formula construction for PARITY is essentially optimal by showing an $\Omega(n^2)$ lower bound on the size of formulas which compute PARITY. In contrast, in a circuit using fan-out larger than one we can expand these XOR gates into AND and OR gates and negations while keeping the number of gates in the circuit linear.

Besides being a testing grounds for circuit techniques, proving lower bounds on formula size also has complexity theoretic consequences as the class NC^1 corresponds to the set of languages computable by polynomial size formulas. The current best formula size lower bound for an explicit function is $n^{3-o(1)}$ by Håstad [Hås98].

In this chapter, we give a fairly complete overview of known techniques for proving formula size lower bounds. In the next chapter, we develop a new framework for proving formula size lower bounds, and more generally lower bounds on communication complexity, using spectral methods. With this technique we do not break the n^3 formula size barrier. We are able, however, to generalize several methods from the literature, in some cases doing provably better, and provide a common framework for viewing some seemingly very different techniques. Perhaps most interestingly, our results indicate a connection between the formula size of a function and its complexity in a very different model, quantum query complexity. In particular, our results give some support to the provocative conjecture that the square of the quantum query complexity of a function f lower bounds its formula size.

8.2 Preliminaries

8.2.1 Deterministic and probabilistic formulae

A Boolean formula over the standard basis $\{\vee, \wedge, \neg\}$ is a binary tree where each internal node is labelled with $\vee$ or $\wedge$, and each leaf is labelled with a literal, that is, a Boolean variable or its negation. The size of a formula is its number of leaves. The depth of a formula is the length of the longest path from a leaf to the root. We naturally identify a formula with the function it computes.

8.2.1. Definition. Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. The formula size of f, denoted $\mathsf{L}(f)$, is the size of the smallest formula which computes f. The formula depth of f, denoted $\mathsf{d}(f)$ is the minimum depth of a formula computing f.

It is clear that $\mathsf{L}(f) \leq 2^{\mathsf{d}(f)}$; that in fact the opposite inequality $\mathsf{d}(f) \leq O(\log \mathsf{L}(f))$ also holds is a nontrivial result due to Spira [Spi71].

We will also consider probabilistic formulae, which we take as a probability distribution over deterministic formulae. We consider a worst-case notion of the size of a probabilistic formula. This model of probabilistic formulae has been studied in the series of works [Val84, Bop89, DZ97] which investigate constructing efficient deterministic monotone formulae for the majority function by amplifying the success probability of probabilistic formulae. The interested reader can also compare our definition with two different models of probabilistic formula size considered by [Kla04].

8.2.2. Definition. Let $S \subseteq \{0,1\}^n$ and $\{f_j\}_{j \in J}$ be a set of functions with $f_j : S \to \{0,1\}$ for each $j \in J$. For a function $f : S \to \{0,1\}$, we say that f is ϵ-approximated by $\{f_j\}_{j \in J}$ if there is a probability distribution $\alpha = \{\alpha_j\}_{j \in J}$ over J such that for every $x \in S$,

$$\Pr_{\alpha}[f(x) = f_j(x)] \geq 1 - \epsilon.$$

In particular, if $\max_j \mathsf{L}(f_j) \leq s$, then we say that f is ϵ-approximated by formulas of size s, denoted $\mathsf{L}^{\epsilon}(f) \leq s$.

Note that even if a function depends on all its variables, it is possible that the probabilistic formula size is less than the number of variables. For example, the OR function on n variables can be computed with error ϵ by a probabilistic formula of size $(1-\epsilon)n$. We simply take a uniform distribution over all formulas which compute the OR of a subset of $(1-\epsilon)n$ many variables. It is clear that the formula is always correct if the input contains no ones. On the other hand say that $x_i = 1$. Then our success probability is the probability that i is included in a random subset of n of size $(1-\epsilon)n$, which is $1-\epsilon$.

8.2.2 Complexity measures of Boolean functions

We use standard complexity measures of Boolean functions, such as sensitivity and certificate complexity (see [BW02] for a survey).

8.2.3. Definition (Sensitivity). *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$ be a Boolean function. For an input $x \in S$ the sensitivity of f on x is the number of strings y which differ from x in exactly one position and such that $f(x) \neq f(y)$. The* zero-sensitivity *of f, written $s_0(f)$ is the maximum over all $x \in f^{-1}(0)$ of the sensitivity of f on x. The* one-sensitivity*, written $s_1(f)$, is defined analogously. Finally, the* sensitivity *of f, written $s(f)$ is the maximum of $s_0(f), s_1(f)$.*

8.2.4. Definition (Certificate Complexity). *Let f be as above. A* certificate *for f on input $x \in S$ is a subset $I \subseteq [n]$ such that for any y satisfying $y_i = x_i$ for all $i \in I$ it must be the case that $f(y) = f(x)$. The* certificate size *of f on input x is the size of a smallest certificate for f on x. The* zero-certificate complexity *of f, written $C_0(f)$ is the maximum over all $x \in f^{-1}(0)$ of the certificate size of f on x. The* one-certificate complexity *of f, is defined analogously. Finally, the* certificate complexity *of f, written $C(f)$, is the maximum of $C_0(f), C_1(f)$.*

8.2.3 Linear Algebra

We will make extensive use of linear algebra in this part of the thesis. Notation and definitions are given in Appendix B.

8.3 Shrinkage Exponent of Boolean Formulae

We first review the source of the best currently known formula size lower bounds. Take a formula ϕ over the basis $\{\wedge, \vee, \neg\}$. A random restriction with probability p leaves each variable x_i free with probability p and sets it to 0 or 1 each with probability $(1-p)/2$. How much do we expect ϕ to shrink after we hit it with such a random restriction and simplify the resulting formula? An obvious answer is that the formula will shrink by a factor of at least p. Subbotovskaya was the first to notice that formulae actually shrink more [Sub61]. She observed that if the original formula size was L, the expected formula size after the random restriction is $O(p^{1.5}L+1)$. The shrinkage exponent Γ is defined as the least upper bound which can replace 1.5 in the previous expression. As the PARITY function shrinks by a factor $\theta(p^2)$, it follows that $\Gamma \leq 2$.

The shrinkage exponent can be used to prove lower bounds on formula size in the following way. Suppose that we have a formula ϕ and we know that after hitting ϕ with a random restriction with probability p, the resulting formula ϕ' still computes with high probability a function requiring formula size L. Well, then the original formula ϕ must have been of size at least L/p^Γ.

Along these lines, Andreev [And87] defined a function $f : \{0,1\}^{2n} \rightarrow \{0,1\}$ whose formula size depends on the shrinkage exponent. On input x, interpret the first n bits of x as the truth table of a function $g : \{0,1\}^{\log n} \rightarrow \{0,1\}$. Divide the second half of x into $\log n$ many blocks $y_1, \ldots, y_{\log n}$ each of length $n/\log n$. Then $f(x) = g(\oplus y_1, \ldots, \oplus y_{\log n})$.

We can show that f requires large formula size as follows. Fix the first half of the input to some hard function g requiring formula size $\Omega(n/\log n)$ which we know to exist by a counting argument. Fixing the function f in this way may decrease its formula size, so say it now has formula size L. Now hit the second half of the input with a random restriction with $p = \frac{\log n \log\log n}{n}$. Say the formula size after this restriction is L'. With this choice of p, with high probability there will be one variable which is not fixed in each of the $\log n$ blocks and thus in particular the resulting formula must still compute g. Thus $L' \geq n/\log n$. Now we can lower bound the size of L as $L' \leq p^\Gamma L$, which implies $L \geq n^{1+\Gamma-o(1)}$.

Håstad shows that the shrinkage exponent is 2, and thus obtains an $n^{3-o(1)}$ formula size lower bound for an explicit function [Hås98]. This proof is quite technical, and we will not pretend to survey it here. We will, however, isolate one key lemma from Håstad's proof which we will see later is in fact a special case of our techniques.

8.3.1. Definition. For any function $f : \{0,1\}^n \rightarrow \{0,1\}$,

1. A *restriction* is a string in $\{0,1,\star\}^n$ where $\star$ means the variable is left free, and 0 or 1 mean the variable is set to the constant 0 or 1, respectively.
2. The restricted function $f|_\rho$ is the function that remains after the non-$\star$ variables in ρ are fixed.
3. R_p is the distribution on random restrictions to the variables of f obtained by setting each variable, independently, to $\star$ with probability p, and to 0 or 1 each with probability $\frac{(1-p)}{2}$.
4. A *filter* Δ is a set of restrictions which has the property that if $\rho \in \Delta$, then every ρ' obtained by fixing one of the $\star$s to a constant is also in Δ.
5. When p is known from the context, and for any event E, and any filter Δ, we write $\Pr[E|\Delta]$ to mean $\Pr_{\rho \in R_p}[E|\rho \in \Delta]$.

8.3.2. Theorem (Håstad, Lemma 4.1). *Let $f : \{0,1\}^n \rightarrow \{0,1\}$. Let A be the event that a random restriction in R_p reduces f to the constant 0, B be the event that a random restriction in R_p reduces f to the constant 1, and let C be the event that a random restriction $\rho \in R_p$ is such that $f|_\rho$ is a single literal. Furthermore, let Δ be any filter. Then*

$$\mathsf{L}(f) \geq \frac{\Pr[C|\Delta]^2}{\Pr[A|\Delta]\Pr[B|\Delta]} \left(\frac{1-p}{2p}\right)^2$$

8.4 Hamming distance one methods

We now discuss two methods, the methods of Khrapchenko and Koutsoupias which relate the formula size of f to the number of (x,y) pairs where $f(x) \neq f(y)$ and x, y have Hamming distance one. We will see in the next chapter that our technique generalizes these methods in a natural way to the case where (x,y) do not necessarily have Hamming distance one.

8.4.1 Khrapchenko's method

Khrapchenko developed a method to give the first tight lower bound of $\Omega(n^2)$ for the parity function [Khr71]. The method works as follows. Let f be a Boolean function, and let X be the set of inputs for which f evaluates to 0, and Y be the set of inputs for which f evaluates to 1. Now think of a bipartite graph with left hand side X, and right hand side Y, and connect two vertices (x,y) by an edge if they have Hamming distance one. The theorem of Khrapchenko says that the product of the average degree of the left hand side in this graph and the average degree of the right hand side is a lower bound on formula size. It is easy to see that for parity every vertex has degree n, thus we get the n^2 lower bound.

8.4.1. Theorem (Khrapchenko). *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$. Let $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$. Let C be the set of pairs $(x,y) \in A{\times}B$ with Hamming distance one, that is $C = \{(x,y) \in A{\times}B : d_H(x,y) = 1\}$. Then $\mathsf{L}(f) \geq \frac{|C|^2}{|A||B|}$.*

8.4.2 Koutsoupias' Method

Koutsoupias extended Khrapchenko's method with a spectral technique [Kou93]. This technique is still restricted to look at pairs which disagree on the function and have Hamming distance one.

8.4.2. Theorem (Koutsoupias). *Let $f : \{0,1\}^n \to \{0,1\}$, and let $X = f^{-1}(0)$, and $Y = f^{-1}(1)$. Let $C = \{(x,y) \in A{\times}B : d_H(x,y) = 1\}$. Let Q be a $|X|{\times}|Y|$ matrix $Q[x,y] = C(x,y)$ where C is identified with its characteristic function. Then $\mathsf{L}(f) \geq \|Q\|^2$.*

8.5 Communication Complexity

8.5.1 Karchmer–Wigderson Game

Karchmer and Wigderson [KW88] give an equivalent formulation of the formula size of f in terms of the communication complexity of a relation associated with f. Since this seminal paper, most results about formula size have been cast in the more general setting of communication complexity. This turns out to have several advantages, in particular, access to the combinatorial view of communication complexity based on rectangles.

Let X, Y, Z be finite sets, and $R \subseteq X{\times}Y{\times}Z$. In the communication game for the relation R, Alice is given some $x \in X$, Bob is given some $y \in Y$ and their goal is to find some $z \in Z$ such that $(x,y,z) \in R$, if such a z exists. A communication protocol is a binary tree where each internal node v is labelled by a function $a_v : X \to \{0,1\}$ or $b_v : Y \to \{0,1\}$ describing either Alice's or Bob's message at that node, and where each leaf is labelled with an element $z \in Z$. A communication protocol computes R if for all $(x,y) \in X{\times}Y$ walking down the tree according to a_v, b_v leads to a leaf labelled with z such that $(x,y,z) \in R$, provided such a z exists. The communication cost $\mathsf{D}(R)$ of R is the depth of the smallest communication protocol computing R. The communication matrix of R is the matrix $M_R[x,y] = \{z : R(x,y,z)\}$. A rectangle $X'{\times}Y'$ with $X' \subseteq X$ and $Y' \subseteq Y$ is monochromatic if $\bigcap_{x \in X', y \in Y'} M_R[x,y] \neq \emptyset$. The protocol partition number $C^P(R)$ is the number of leaves in the smallest communication protocol computing R, and the rectangle partition number $C^D(R)$ is the smallest number of disjoint monochromatic rectangles required to cover $X{\times}Y$. We will often informally refer to $C^D(R)$ as the "rectangle bound".

8.5.1. Definition. Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$. We associate with f a relation $R_f = \{(x, y, i) : f(x) = 0, f(y) = 1, x_i \neq y_i\}$.

8.5.2. Theorem (Karchmer-Wigderson). *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$ be a Boolean function. Then $\mathsf{L}(f) = C^P(R_f)$.*

Proof: We first see the direction $C^P(R_f) \leq \mathsf{L}(f)$. Let ϕ be a formula for f of size $\mathsf{L}(f)$, and assume without loss of generality that ϕ only has negations at its leaves. We use ϕ to define our communication protocol. Say that Alice has $x \in f^{-1}(0)$ and Bob has $y \in f^{-1}(1)$. Initially, $\phi(x) = 0$ and $\phi(y) = 1$. We move down the formula tree choosing subformulas which continue to satisfy these conditions. Thus for example if node v is labelled by an AND gate, say the AND of subformulas ϕ_{v0}, ϕ_{v1}, then Alice will speak according to the rule $a_v(x) = 0$ if $\phi_{v0}(x) = 0$ and $a_v(x) = 1$ otherwise. Continuing in this manner with Alice speaking at the AND gates and Bob speaking at the OR gates, we will eventually arrive at a leaf labeled by a single literal ℓ_i such that $\ell_i(x) \neq \ell_i(y)$. Thus it must be the case that $x_i \neq y_i$, and so we label this leaf with index i. It is clear that the number of leaves of the communication protocol so defined is simply $\mathsf{L}(f)$, and the communication cost is the depth of the formula.

Now for the direction $\mathsf{L}(f) \leq C^P(R_f)$. We prove by induction on the size of the communication protocol. If the communication protocol for R_f has size 1 then there exists some index i such that $x_i \neq y_i$ for all $x \in X$ and $y \in Y$. There is similarly a size 1 formula for f, namely either x_i or $\neg x_i$.

For the induction step, suppose that the statement holds for all relations with protocols of size t, and consider the relation R_f with protocol size $t + 1$. One player speaks first in the protocol, suppose it is Alice. The other case follows similarly. Let X_0 be the set of inputs for which Alice first says 0, and let X_1 be the set of inputs for which Alice first says 1. Let f_j be such that $X_j \subseteq f_j^{-1}(0)$ and $Y \subseteq f_j^{-1}(1)$ for $j \in \{0,1\}$. Now inputs from the sets $X_0 \times Y$ and $X_1 \times Y$ can be solved with by protocols of size t or less, and so by the induction hypothesis there are functions f_j for $j \in \{0,1\}$ with $X_j \subseteq f_j^{-1}(0)$ and $Y \subseteq f_j^{-1}(1)$, and which have formula size t or less. We now set the formula for f to be $f_0 \wedge f_1$. □

8.6 Rectangle Based Methods

Using the Karchmer–Wigderson game, we can turn the problem of lower bounding formula size into lower bounding communication complexity, and thus have at our disposal the well-developed framework for proving communication complexity lower bounds.

One of the most basic facts about communication complexity is that a deterministic protocol for a relation $R \subseteq X \times Y \times Z$ partitions $X \times Y$ into monochromatic rectangles. This means that, if $C^D(R)$ is the size of the smallest partition of R into monochromatic rectangles, then $C^D(R)$ is a lower bound on $C^P(R)$. We now give a proof of this important fact.

8.6.1. Theorem. $C^D(R) \leq C^P(R)$

Proof: For a node v in the protocol tree, let $S_v \subseteq X \times Y$ be the set of inputs which reach v during the protocol. We prove by induction that S_v is a rectangle.

If v is the root, then S_v is all of $X \times Y$ and so a rectangle. Otherwise, let w be the parent of v and suppose, without loss of generality, that v is the left son of w and that Alice speaks at v. By the induction hypothesis, the set of inputs reaching w form a rectangle $X_w \times Y_w$. The set of inputs which reach v then is the set

$$(X_w \cap \{x \in X_w : a_w(x) = 0\}) \times Y_w$$

which is a rectangle.

Thus we have shown that the set of inputs reaching any node in the protocol form a rectangle, thus in particular the inputs reaching a leaf node form a rectangle. Now let v be a leaf node, and suppose that the output at v is $z \in Z$. Notice that if the protocol is correct, then it must be the case that $(x, y, z) \in R$ for all $(x, y) \in S_v$, and so v is monochromatic. Thus each leaf is a monochromatic rectangle.

Furthermore, as each node of the protocol is labelled by a function, and each (x, y) pair lies in some leaf, the leaves form a disjoint covering of $X \times Y$ by monochromatic rectangles.

Now, as the protocol partitions $X \times Y$ into R-monochromatic rectangles in a particular way, it will in turn be lower bounded by the size of the smallest R-monochromatic partition of $X \times Y$, which is $C^D(R)$. □

We have now seen that the rectangle bound can be used to lower bound the size of a smallest communication protocol for a relation R. We now see that this lower bound is never too bad.

8.6.2. Theorem ([KKN95] following [AUY83]). *For any relation R,*

$$C^D(R) \leq C^P(R) \leq 2^{O(\log^2 C^D(R))}$$

Proof: Let $R \subseteq X \times Y \times Z$ be a relation, and $\mathcal{R}$ be a rectangle partition of R of size $C^D(R)$. Using the partition $\mathcal{R}$, we design a protocol with communication complexity $O(\log^2 C^D(R))$.

Initially, there are at most $C^D(R)$ many rectangles which are “alive”, that is which could contain (x, y). The protocol will proceed in phases, where in each phase Alice and Bob communicate $\log C^D(R) + O(1)$ bits and reduce the number of alive rectangles by a factor of 2. Thus after at most $\log C^D(R)$ many phases, they have reduced the number of alive rectangles to 1, and output the color of this rectangle.

The crucial fact we use is that two disjoint rectangles cannot intersect in both rows and columns. This means that if we look at the rectangle S containing (x, y) then the rectangles which are still alive and intersect S in columns must be distinct from the rectangles which are still alive and intersect S in rows. This gives a way to reduce the number of alive rectangles under consideration in half. We now describe in more detail the content of each phase:

1. Alice considers all the rectangles which are still alive. If there is only 1 rectangle still alive, then she outputs the color of this rectangle. Otherwise, she searches for a rectangle S which intersects x and which intersects in rows with at most half of the rectangles still alive. If she finds such a rectangle she tells Bob its name and the phase is completed. Otherwise, she tells Bob no such rectangle exists.

2. Bob searches for a rectangle which is still alive, contains the column y, and intersects in columns with at most half of the rectangles still alive. By the reasoning above, such a rectangle must exist. Bob sends its name to Alice and the phase is completed.

We see that each phase requires at most $\log C^D(R) + O(1)$ many bits and reduces the number of alive rectangles in half. Thus the total communication complexity is $O(\log^2 C^D(R))$. □

8.6.1 Rank Method

Razborov proposed a method for proving formula size lower bounds based on matrix rank. The key property underlying this method is that matrix rank is subadditive, that is $\mathrm{rk}(A + B) \leq \mathrm{rk}(A) + \mathrm{rk}(B)$. One disadvantage with the rectangle bound is that one has to evaluate a minimum over all monochromatic rectangle partitions, which can be an doubly exponentially large set, in terms of the input length n of the function. For proving formula size lower bounds, Razborov gets around this problem by noticing there is a small set of n rectangles, which he calls the canonical cover, such that every monochromatic rectangle is contained in some rectangle from the canonical cover.

8.6.3. Definition. Let $R \subseteq X \times Y \times Z$ be a relation. The *canonical cover* of R is the set system $\{S_z\}_{z \in Z}$, where

$$S_z = \{(x, y) : x \in X, y \in Y, (x, y, z) \in R\}.$$

For a matrix A with rows labelled from a set X and columns labelled from a set Y, and a set $S \subseteq X \times Y$, let A_S be the matrix A with all entries not in the set S set to 0.

8.6.4. Theorem (Razborov).

$$C^D(R) \geq \max_{A \neq 0} \frac{\mathrm{rk}(A)}{\max_z \mathrm{rk}(A_{S_z})}$$

Proof: Let $\mathcal{R}$ be a monochromatic partition of R such that $C^D(R) = |\mathcal{R}|$. Let $A \neq 0$ be a matrix over $X \times Y$. By subadditivity of rank, we have $\mathrm{rk}(A) \leq \sum_{S \in \mathcal{R}} \mathrm{rk}(A_S) \leq C^D(R) \max_{S \in \mathcal{R}} \mathrm{rk}(A_S)$. Now notice that for any $S \in \mathcal{R}$, as S is monochromatic it must be a subset of an element of the canonical cover S_z for some z. As S is a rectangle, it now follows that $\mathrm{rk}(A_S) \leq \mathrm{rk}(A_{S_z})$, as a set of independent vectors for A_S will remain independent by adding entries which are disjoint from S in either rows or columns. This gives the theorem. □

Razborov shows that his method cannot prove lower bounds larger than $O(n)$ for formula size. For monotone formula size, however, he uses this technique to prove $n^{\Omega(\log n)}$ lower bounds.

8.6.2 Rectangle Size Technique

Say we are looking at a monochromatic rectangle partition of a set $X \times Y$. An obvious lower bound on the size of such a partition is

$$\frac{|X||Y|}{\max_S |S|}$$

where S ranges over all monochromatic rectangles. One can generalize this bound by considering instead probability distributions over $X \times Y$.

8.6.5. Definition. Let $R \subseteq X \times Y \times Z$ be a relation, and let μ be a probability distribution on $X \times Y$. Define

$$N^*(R) = \max_{\mu \neq 0} 1 / \max_S \mu(S)$$

where S ranges over all monochromatic rectangles of R, and $\mu(S) = \sum_{(x,y) \in S} \mu(x, y)$.

It is clear that $N^*(R) \leq C^D(R) \leq C^P(R)$. Thus far, we have talked about the smallest monochromatic rectangle partition of a relation. One can also consider the smallest monochromatic rectangle *covering*, denoted $C^N(R)$. We use this notation as $\log C^N(R)$ is equal to the nondeterministic communication complexity of R. Karchmer, Kushilevitz, and Nisan [KKN95] show that in turn $\log N^*(R)$ characterizes nondeterministic communication complexity:

8.6.6. Theorem ([KKN95]).

$$\log N^*(R) \leq \log C^N(R) \leq \log N^*(R) + \log n + O(1)$$

8.6.3 Linear programming bound

In the same paper, Karchmer, Kushilevitz, and Nisan [KKN95] introduce another technique based on a linear programming approximation of the rectangle bound. We will refer to this as the KKN bound. They first observe that the rectangle bound can be written as a 0-1 integer program. Let R be a relation, and $\mathcal{S}$ be the set of all rectangles which are monochromatic with respect to R. Consider the matrix A with rows labelled by elements of $X \times Y$ and columns labelled by rectangles from $\mathcal{S}$. Then the rectangle bound is exactly equal to the following optimization problem:

$$\begin{array}{l} \text{minimize } \sum_i x_i \\ \text{such that } Ax = 1 \quad \text{and} \quad x_i \in \{0,1\} \text{ for all } i. \end{array}$$

We can relax this into a linear program as follows:

$$\begin{array}{l} \text{minimize } \sum_i x_i \\ \text{such that } Ax = 1 \quad \text{and} \quad x \geq 0. \end{array}$$

The bound of the KKN technique is then the optimum of this program. More useful for proving formula size lower bounds is the dual of this program:

$$\begin{array}{l} \text{maximize} \sum_{x,y} w(x,y) \\ \text{such that } \sum_{x,y \in S} w(x,y) \leq 1 \text{ for every } S \in \mathcal{S}. \end{array}$$

Notice that the essential difference between this method and the rectangle size technique is that here the weight function can be negative. KKN use this technique to reprove Khrapchenko's result that the formula size of PARITY is $\Theta(n^2)$. They also show, however, that this technique cannot prove lower bounds larger than n^2.

8.7 Nečiporuk

Nečiporuk's method [Neč66] is somewhat different from the aforementioned methods. For one, it works over any basis of fan-in 2 functions, not just $\{\wedge, \vee, \neg\}$. Klauck has shown a close connection between the Nečiporuk method and one-way communication complexity [Kla04].

Let f be a function on the n Boolean variables $X = \{x_1, \ldots, x_n\}$. Let $S \subseteq X$. An S-subfunction of f is the function obtained by fixing the variables in $X - S$ to some value.

8.7.1. Theorem (Nečiporuk). *Let f be a function on n variables and $S_1, \ldots, S_k$ be a partition of the variables. Say that f has s_i many distinct S_i-subfunctions. Then*

$$\mathsf{L}(f) \geq \frac{1}{4} \sum_{i=1}^{k} \log s_i.$$

The largest lower bounds provable with Nečiporuk's method are of the order $\Theta(n^2/\log n)$.

8.8 Summary of our Results

We have now seen nearly all of the general techniques which have been developed to prove formula size lower bounds. In Chapter 9, we develop a new algebraic method for proving formula size lower bounds based on matrix spectra. To prove that our method lower bounds formula size we use the Karchmer–Wigderson characterization of formula size as a communication complexity game, as seen in Section 8.5.1. This allows us to show that our method can in general be used to lower bound $C^D(R)$, the size of the smallest monochromatic rectangle partition of the relation R, and thus also communication complexity.

Our spectral methods can be seen as a general framework which encompasses all of the Hamming distance one methods discussed in Section 8.4, and also the rectangle size technique discussed in Section 8.6.2. In Chapter 10, we look at a special case of our method in detail. This method has orgins from the so-called quantum adversary method, one of two main techniques used to prove lower bounds on quantum query complexity. We show that the square of the quantum adversary method gives a lower bound on formula size. In this case, we are able to give concrete examples of functions where our new method gives provably stronger bounds than the Hamming distance one methods of Khrapchenko and Koustsoupias, and the lemma of Håstad, Theorem 8.3.2.

Chapter 9

Spectral Methods for Formula Size Lower Bounds

9.1 Introduction

In this chapter, we develop a new technique for proving formula size lower bounds based on spectral methods. To introduce this technique, we first recall from the previous chapter the probability on rectangles method of [KKN95]. For a relation $R \subseteq X \times Y \times Z$, one puts a probability distribution μ on $X \times Y$ and the bound given is

$$\max_{\mu \neq 0} \frac{1}{\max_S \mu(S)} \tag{9.1}$$

where S ranges over all monochromatic rectangles of R.

Now let us take a look at Equation (9.1) from a different point of view. Recall that the Frobenius norm of a matrix A, written $\|A\|_F$, is $\sqrt{\sum_{ij} |A[i,j]|^2}$. One can easily verify that the Equation (9.1) is equivalent to

$$\max_{A \neq 0} \frac{\|A\|_F^2}{\max_S \|A_S\|_F^2}$$

where again S ranges over all monochromatic rectangles of R and A_S is the matrix A with entries outside of S set to 0.

We now further rewrite this expression. First we need a bit of notation. For a square $n \times n$ matrix A, let $\lambda_1(A) \geq \ldots \geq \lambda_n(A)$ be the eigenvalues of A. For an arbitrary matrix A, let $\sigma_i(A) = \sqrt{\lambda_i(A^*A)}$, be the ith singular value of A. Now notice that $\sum_{ij} |A[i,j]|^2 = \mathrm{Tr}(A^*A) = \sum_{i=1}^n \sigma_i^2(A)$. Thus Equation (9.1) is equivalent to

$$\max_{A \neq 0} \frac{\sum_{i=1}^n \sigma_i^2(A)}{\max_S \sum_{i=1}^n \sigma_i^2(A_S)}.$$

We show that this is only the beginning of spectral based bounds in communication complexity:

9.1.1. Theorem. *Let $R \subseteq X \times Y \times Z$, and let $N = \min\{|X|, |Y|\}$. Then for any $1 \leq k \leq N$*

$$C^P(R) \geq \max_{A \neq 0} \frac{\sum_{i=1}^k \sigma_i^2(A)}{\max_S \sum_{i=1}^k \sigma_i^2(A_S)}.$$

where S ranges over all rectangles monochromatic with respect to R.

Of particular interest is the case $k = 1$. In this case, Theorem 9.1.1 is related to the square of the so-called quantum adversary method, a method developed for proving lower bounds on quantum query complexity which has been given many alternative formulations [Amb02, Amb03, BSS03, LM04, Zha05] which all turn out to be equivalent [ŠS05].

In this chapter we prove Theorem 9.1.1, and keep our discussion at a somewhat abstract level; in the following chapter, we analyze in detail applications of this theorem in the case $k = 1$. We will see that this method subsumes the Hamming distance one methods discussed earlier, and also look at the application of this method in some concrete cases.

9.2 The proof

To prove Theorem 9.1.1 we first formulate a lemma which says that the sum of the squares of the first k singular values is subadditive on matrices B, C if B and C are disjoint on rows. We actually state the lemma in a bit more generality—we feel this is justified as it highlights exactly the condition which is needed in the proof.

9.2.1. Definition. We say that two matrices B, C are orthogonal to each other if Bx and Cx are orthogonal as vectors for all vectors x.

9.2.2. Lemma. *Let A be a $m \times n$ matrix with $n \leq m$. Let B, C be orthogonal to each other such that $A = B + C$. Then for all $1 \leq k \leq n$,*

$$\sum_{i=1}^k \sigma_i^2(A) \leq \sum_{i=1}^k \sigma_i^2(B) + \sigma_i^2(C).$$

The main difficulty of the lemma is obtaining the proper characterization of the sum of the first k singular values. To this end, we need the following theorem of Ky Fan [Fan49]. We give a new proof of this theorem here which seems simpler than the original proof of Ky Fan, or the “simplified” proof given by [OW92].

9.2.3. Theorem (Ky Fan). *Let A be a $n \times n$ Hermitian matrix, and let $\lambda_1 \geq \cdots \geq \lambda_n$ be the eigenvalues of A. Then for all $1 \leq k \leq n$,*

$$\sum_{i=1}^{k} \lambda_i = \max_{X:X^*X=I} \mathrm{Tr}(X^*AX) \tag{9.2}$$

where X ranges over $n \times k$ matrices.

Proof: One direction of the inequality is clear: Let the columns of X be orthonormal eigenvectors corresponding to $\lambda_1, \ldots, \lambda_k$, respectively. Then $\mathrm{Tr}(X^*AX) = \sum_i \lambda_i$ and so the right hand side of Equation (9.2) is larger than the left hand side.

The other direction takes a bit more work. As A is Hermitian, there is a unitary matrix U and a diagonal matrix $\Lambda = \mathrm{diag}[\lambda_1, \ldots, \lambda_n]$ such that $A = U^*\Lambda U$. We now have:

$$\begin{aligned}
\max_{X:X^*X=I} \mathrm{Tr}(X^*AX) &= \max_{X:X^*X=I} \mathrm{Tr}(X^*U^*\Lambda UX) \\
&= \max_{X:X^*X=I} \mathrm{Tr}((UX)^*\Lambda(UX)) \\
&= \max_{X:X^*X=I} \mathrm{Tr}(X^*\Lambda X) \\
&= \max_{X:X^*X=I} \sum_{i=1}^{k} \sum_{j=1}^{n} \lambda_j |X[i,j]|^2 \\
&= \max_{X:X^*X=I} \sum_{j=1}^{n} \lambda_j \sum_{i=1}^{k} |X[i,j]|^2.
\end{aligned}$$

To complete the proof, we now show that $\sum_{i=1}^{k} |X[i,j]|^2 \leq 1$ for every j. To see this, extend X to a $n \times n$ unitary matrix Y whose first k columns are the same as those of X. It then follows that $\sum_{i=1}^{k} |X[i,j]|^2 \leq (YY^*)[j,j] = 1$. Let $\alpha_j = \sum_{i=1}^{k} |X[i,j]|^2$ be the "weight" on λ_j. Notice that the weight on each λ_j is at most 1, that is $\alpha_j \leq 1$, and in total we have k units of weight to distribute, that is $\sum_{j=1}^{n} \alpha_j = k$. We now find

$$\max_{X:X^*X=I} \sum_{j=1}^{n} \lambda_j \sum_{i=1}^{k} |X[i,j]|^2 = \sum_{j=1}^{n} \alpha_j \lambda_j \tag{9.3}$$

As we have k units of weight to distribute, it is best to concentrate them on the first k largest eigenvalues. Further, as each eigenvalue can receive at most one unit of weight the best we can do is to let $\alpha_i = 1$ for each $1 \leq i \leq k$. Here we achieve the value $\sum_{i=1}^{k} \lambda_i$. □

Proof: [of Lemma 9.2.2] Notice that A^*A is a Hermitian matrix and that $\lambda_i(A^*A) = \sigma_i^2(A)$. Thus by the Ky Fan theorem we have

$$\begin{aligned}\sum_{i=1}^k \sigma_i^2(A) &= \max_{X:X^*X=I} \mathrm{Tr}(X^*A^*AX) \\ &= \max_{X:X^*X=I} \sum_{i=1}^k \|AX_i\|^2\end{aligned}$$

where X_i is the ith column of X. With this characterization in hand, the proof is now easy:

$$\begin{aligned}\sum_{i=1}^k \sigma_i^2(A) &= \max_{X:X^*X=I} \sum_{i=1}^k \|AX_i\|^2 \\ &= \max_{X:X^*X=I} \sum_{i=1}^k \|BX_i + CX_i\|^2 \\ &= \max_{X:X^*X=I} \sum_{i=1}^k \|BX_i\|^2 + \|CX_i\|^2 \\ &\leq \max_{X:X^*X=I} \sum_{i=1}^k \|BX_i\|^2 + \max_{X:X^*X=I} \sum_{i=1}^k \|CX_i\|^2 \\ &= \sum_{i=1}^k \sigma_i^2(B) + \sigma_i^2(C).\end{aligned}$$

where in the third line we used the fact that B, C are orthogonal to each other. □

We now extract a sufficient condition for a matrix function to lower bound communication complexity.

9.2.4. Lemma. *Let μ be a function from matrices to nonnegative real numbers. Suppose that μ satisfies the following three conditions:*

1. *$\mu(A) = 0$ if and only if $A = 0$.*

2. *If $A = B + C$ and B, C are disjoint on rows, then $\mu(A) \leq \mu(B) + \mu(C)$.*

3. *If $A = B + C$ and B, C are disjoint on columns, then $\mu(A) \leq \mu(B) + \mu(C)$.*

4. *if S is a rectangle, and $S \subseteq S'$ then $\mu(A_S) \leq \mu(A_{S'})$.*

Then for any relation $R \subseteq X \times Y \times Z$

$$C^P(R) \geq \max_{A \neq 0} \frac{\mu(A)}{\max_S \mu(A_S)}$$

where S ranges over all the monochromatic rectangles of R.

Proof: Fix a matrix $A \neq 0$. Let P be a protocol for R whose size is $C^P(R)$. For a node v in the protocol tree, let $S_v \subseteq X \times Y$ be the set of elements which reach v in the protocol. We will abuse notation by writing A_v for A_{S_v}.

We prove by induction on the depth of the protocol the following statement:

$$\mu(A) \leq \sum_{\text{leaves } \ell} \mu(A_\ell).$$

Say that Alice speaks first (and that Alice's inputs label the rows). Let A_0 consist of those rows of A labeled by inputs where Alice says 0 and A_1 consist of those rows of A where Alice says 1. Then $A = A_0 + A_1$, and by the second property of the μ, we have $\mu(A) \leq \mu(A_0) + \mu(A_1)$. If instead Bob speaks first, then similarly we can define A_0 and A_1 which are disjoint on columns, and then use the third property.

Now suppose the statement is true up to depth k, that is

$$\mu(A) \leq \sum_v \mu(A_v).$$

where the sum is over all nodes at depth k. Consider now some node v at depth k, and suppose that Alice speaks at this node. Then notice that $A_v = A_{v0} + A_{v1}$ and that A_{v0} and A_{v1} are disjoint on rows. Thus by property 2, the sum does not decrease by this subdivision. The other case follows similarly using property 3.

It thus follows that

$$\mu(A) \leq \sum_{\text{leaves } \ell} \mu(A_\ell) \leq C^P(R) \max_\ell \mu(A_\ell). \tag{9.4}$$

We know that in a successful protocol each leaf ℓ is a monochromatic rectangle, thus by property 3 and the fact that $A \neq 0$ was arbitrary, we obtain the statement of the lemma. □

We would like to highlight the role of property 4 in this proof. Notice that we did not use property 4 in arriving to Equation (9.4). Evaluating this equation, however, would require knowing the structure of the optimal protocol. If this were known, we would not be messing around with μ in trying to prove lower bounds. We use property 4, therefore, to go from a lower bound which requires knowledge of the optimal protocol for R to one which relies only on properties of R itself, namely its structure of monochromatic rectangles.

Proof:[of Theorem 9.1.1] We only need to verify that $\sum_{i=1}^k \sigma_i^2(A)$ satisfies the four properties of Lemma 9.2.4. The first property is clear. The second property is given by Lemma 9.2.2. The third property follows by applying Lemma 9.2.2 and the fact that A^*A and AA^* have the same nonzero eigenvalues (see Chapter B) and so $\sum_{i=1}^k \sigma_i^2(A) = \sum_{i=1}^k \sigma_i^2(A^*)$.

Property 4 deserves more comment. Recall that Property 4 states: If S is a rectangle and $S \subseteq S'$ then $\mu(A_S) \leq \mu(A_{S'})$. This property follows from the interlacing theorem for singular values which can be found in [HJ99], Theorem 7.3.9:

9.2.5. Theorem. *Let A be a m-by-n matrix and let $\hat{A}$ be the matrix obtained by deleting any one column from A.*

- *If $m \geq n$ then*
$$\sigma_1(A) \geq \sigma_1(\hat{A}) \geq \sigma_2(A) \geq \sigma_2(\hat{A}) \geq \ldots \geq \sigma_{n-1}(\hat{A}) \geq \sigma_n(A) \geq 0.$$
- *If $m < n$ then*
$$\sigma_1(A) \geq \sigma_1(\hat{A}) \geq \sigma_2(A) \geq \sigma_2(\hat{A}) \geq \ldots \geq \sigma_m(A) \geq \sigma_m(\hat{A}) \geq 0.$$

As S is a rectangle, we can obtain A_S by deleting rows and columns from $A_{S'}$ and by the interlacing property we have that the singular values of $A_{S'}$ will be componentwise larger than those of A_S.

It is crucial here that S be a rectangle—the statement is not true for general sets S. □

9.3 Discussion

An immediate question which arises is, what other matrix functions satisfy the conditions of Lemma 9.2.4? We have already seen another example in matrix rank. Searching for others, a natural place to look is amongst the matrix norms. An interesting class of matrix norms are those induced by vector norms. A vector norm $\|\cdot\|$ induces a matrix norm by

$$\|A\| = \max_{x:\|x\|=1} \|Ax\|.$$

For example, consider the matrix norm $\|A\|_p$ induced by the ℓ_p vector norm. One can easily see that if B, C are row disjoint matrices, then $\|B+C\|_p^p \leq \|B\|_p^p + \|C\|_p^p$, thus property 2 also holds for these matrix p-norms to the pth power. The problem is that property 3 does not hold for these matrix p-norms.

A further question to ask is if we have gained something by allowing the maximum over all matrices $A \neq 0$ in Theorem 9.1.1 instead of just nonnegative matrices. Are there examples where this gives a provably stronger bound?

Chapter 10

Quantum Adversary Method and Formula Size Lower Bounds

This chapter is based on the paper:

- S. Laplante, T. Lee, and M. Szegedy. The quantum adversary method and classical formula size lower bounds. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, pages 76–90, 2005. Invited to the *Computational Complexity* special issue of selected papers from CCC 2005.

10.1 Introduction

In this chapter we study in depth a special case of the spectral techniques developed in the last chapter. This is the case where $k = 1$ and the bound becomes

$$\max_{A \neq 0} \frac{\|A\|^2}{\max_S \|A_S\|^2}.$$

This bound is especially interesting as it is related to the so-called quantum adversary method, a technique developed for proving lower bounds on quantum query complexity. We see that this bound generalizes the Hamming distance one techniques discussed in Chapter 8 and we look at a concrete example where this bound gives something provably better than these other methods. We also look at one attempt to extend this bound to give stronger results, and the limitations of these approaches.

10.2 Quantum Adversary Method

The roots of the quantum adversary argument can be traced to the hybrid argument of [BBBV97], who use it to show a $\Omega(\sqrt{n})$ lower bound on quantum search. Ambainis developed a more sophisticated adversary argument [Amb02] and later improved this method to the full-strength quantum adversary argument [Amb03]. Further generalizations include Barnum, Saks, and Szegedy [BSS03] with their spectral method and Zhang [Zha05] with his strong adversary method. Laplante and Magniez [LM04] use Kolmogorov complexity to capture the adversary argument in terms of a minimization problem. This line of research culminates in recent work of Špalek and Szegedy [ŠS05] who show that in fact all the methods of [Amb03, BSS03, Zha05, LM04] are equivalent.

10.2.1 Quantum query complexity

As with the classical counterpart, in the quantum query model we wish to compute some function $f : S \to \{0,1\}$, where $S \subseteq \Sigma^n$, and we access the input through queries. The complexity of f is the number of queries needed to compute f. Unlike the classical case, however, we can now make queries in superposition. Formally, a query O corresponds to the unitary transformation

$$O : |i, b, z\rangle \mapsto |i, b \oplus x_i, z\rangle$$

where $i \in [n], b \in \{0,1\}$, and z represents the workspace. A t-query quantum algorithm A has the form $A = U_t O U_{t-1} O \cdots O U_1 O U_0$, where the U_k are fixed unitary transformations independent of the input x. The computation begins in the state $|0\rangle$, and the result of the computation A is the observation of the rightmost bit of $A|0\rangle$. We say that A ϵ-approximates f if the observation of the rightmost bit of $A|0\rangle$ is equal to $f(x)$ with probability at least $1 - \epsilon$, for every x. We denote by $Q_\epsilon(f)$ the minimum query complexity of a quantum query algorithm which ϵ-approximates f.

10.2.2 Adversary formulations

In this section we present some of the various formulations of the adversary method which have been given. Having these different characterizations on hand will make it very convenient to show that the adversary technique subsumes several different methods for proving formula size lower bounds, namely the Hamming distance one techniques discussed in Section 8.4.

We give the primal characterization as our principal definition of sumPI.

10.2.1. Definition (sumPI). *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$ be a Boolean function. For every $x \in S$ let $p_x : [n] \to \mathbb{R}$ be a probability distribution, that is, $p_x(i) \geq 0$ and $\sum_i p_x(i) = 1$. Let $p = \{p_x : x \in S\}$. We define the sum probability of indices to be*

$$\mathsf{sumPI}(f) = \min_p \max_{\substack{x,y \\ f(x) \neq f(y)}} \frac{1}{\sum_{\substack{i \\ x_i \neq y_i}} \sqrt{p_x(i) p_y(i)}}$$

The name sumPI, short for "sum over probability of indices", is taken from this formulation. By the above mentioned series of works it follows that

$$Q_\epsilon(f) = \left(1 - 2\sqrt{\epsilon(1-\epsilon)}\right) \mathsf{sumPI}(f).$$

We will also use two versions of the dual method, both a weight scheme and the spectral formulation. The most convenient weight scheme for us is the "probability scheme", given in Lemma 4 of [LM04].

10.2.2. Definition (Probability Scheme). *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$ be a Boolean function, and $X = f^{-1}(0), Y = f^{-1}(1)$. Let q be a probability distribution on $X \times Y$, and p_A, p_B be probability distributions on X, Y respectively. Finally let $\{p'_{x,i} : x \in X, i \in [n]\}$ and $\{p'_{y,i} : y \in Y, i \in [n]\}$ be families of probability distributions on Y, X respectively. Assume that $q(x,y) = 0$ when $f(x) = f(y)$. Let P range over all possible tuples $(q, p_A, p_B, \{p'_{x,i}\}_{x,i})$ of distributions as above. Then*

$$\mathsf{PA}(f) = \max_P \min_{\substack{x,y,i \\ f(x)\neq f(y), x_i \neq y_i}} \frac{\sqrt{p_A(x)p_B(y)p'_{x,i}(y)p'_{y,i}(x)}}{q(x,y)}$$

We will also use the spectral adversary method.

10.2.3. Definition (Spectral Adversary). *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$ be a Boolean function. Let $X = f^{-1}(0), Y = f^{-1}(1)$. Let $A \neq 0$ be an arbitrary $|X| \times |Y|$ nonnegative matrix. For $i \in [n]$, let A_i be the matrix:*

$$A_i[x,y] = \begin{cases} 0 & \text{if } x_i = y_i \\ A[x,y] & \text{if } x_i \neq y_i \end{cases}$$

Then

$$\mathsf{SA}(f) = \max_{A \neq 0} \frac{\|A\|}{\max_i \|A_i\|}$$

The spectral adversary method was initially defined in [BSS03] for symmetric matrices over $X \cup Y$. The above definition is equivalent: if A is a $X \cup Y$ matrix satisfying the constraint that $A[x,y] = 0$ when $f(x) = f(y)$ then A is of the form $A = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}$, for some matrix B over $X \times Y$. Then the spectral norm of A is equal to that of B. Similarly, for any $X \times Y$ matrix A we can form a symmetrized version of A as above preserving the spectral norm.

We will often use the following theorem implicitly in taking the method most convenient for the particular bound we wish to demonstrate.

10.2.4. Theorem (Špalek-Szegedy). *Let $n \geq 1$ be an integer, $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$. Then*

$$\mathsf{sumPI}(f) = \mathsf{SA}(f) = \mathsf{PA}(f)$$

10.2.3 Properties of sumPI

Besides having such a robust definition, the measure sumPI behaves very nicely under composition. We are able to give both upper and lower bounds on the sumPI measure of a composed function in terms of the sumPI measures of its component functions.

10.2.5. Lemma. *Let $g_1, \ldots, g_n$ be Boolean functions, and h be a function, $h : \{0,1\}^n \to \{0,1\}$. If $\mathsf{sumPI}(g_j) \leq a$ for $1 \leq j \leq n$ and $\mathsf{sumPI}(h) \leq b$, then for $f = h(g_1, \ldots, g_n)$, $\mathsf{sumPI}(f) \leq ab$.*

Proof: Let p be an optimal family of distribution functions associated with h and p_j be optimal families of distribution functions associated with g_j. Define the distribution function

$$q_x(i) = \sum_{j \in [n]} p_{g(x)}(j) p_{j,x}(i).$$

Assume that for $x, y \in S$ we have $f(x) \neq f(y)$. It is enough to show that

$$\sum_{i:\, x_i \neq y_i} \sqrt{\sum_{j \in [n]} p_{g(x)}(j) p_{j,x}(i)} \sqrt{\sum_{j \in [n]} p_{g(y)}(j) p_{j,y}(i)} \geq \frac{1}{ab}. \tag{10.1}$$

By Cauchy–Schwarz, the left hand side of Equation (10.1) is greater than or equal to

$$\begin{aligned} &\sum_{i: x_i \neq y_i} \sum_{j \in [n]} \sqrt{p_{g(x)}(j) p_{j,x}(i)} \sqrt{p_{g(y)}(j) p_{j,y}(i)} \\ &= \sum_{j \in [n]} \left(\sqrt{p_{g(x)}(j) p_{g(y)}(j)} \sum_{i: x_i \neq y_i} \sqrt{p_{j,x}(i) p_{j,y}(i)} \right). \end{aligned} \tag{10.2}$$

By the definition of p_j, we have $\sum_{i: x_i \neq y_i} \sqrt{p_{j,x}(i) p_{j,y}(i)} \geq 1/a$ as long as $g_j(x) \neq g_j(y)$. Thus we can estimate the expression in Equation (10.2) from below by:

$$\frac{1}{a} \sum_{j: g_j(x) \neq g_j(y)} \sqrt{p_{g(x)}(j) p_{g(y)}(j)}.$$

By the definition of p we can estimate the sum (without the $1/a$ coefficient) in the above expression from below by $1/b$, which finishes the proof. □

The following powerful lemma of Ambainis [Amb03] makes it easy to lower bound the sumPI complexity of iterated functions.

10.2.6. Definition. Let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function. We define the d^{th} iteration of f, written $f^d : \{0,1\}^{n^d} \to \{0,1\}$, inductively as $f^1(x) = f(x)$ and

$$f^{d+1}(x) = f(f^d(x_1, \ldots, x_{n^d}), f^d(x_{n^d+1}, \ldots, x_{2n^d}), \ldots, \\ f^d(x_{(n-1)n^d+1}, \ldots, x_{n^{d+1}}))$$

10.2.7. Lemma (Ambainis). *Let f be any Boolean function and f^d the d^{th} iteration of f. Then $\mathsf{sumPI}(f^d) \geq (\mathsf{sumPI}(f))^d$.*

Combining this with Lemma 10.2.5, we get:

10.2.8. Corollary. *Let f be any Boolean function and f^d the d^{th} iteration of f. Then $\mathsf{sumPI}(f^d) = (\mathsf{sumPI}(f))^d$.*

10.3 Key Combinatorial Lemma

As in the previous chapter, the key to our results is a combinatorial lemma relating the square of the spectral norm of a matrix to the sum of the squares of the spectral norms of its submatrices. In the case of the spectral norm, we are able to prove a stronger result than that given in the previous chapter, namely that the spectral norm squared is subadditive with respect to an arbitrary rectangle decomposition, not just a treelike decomposition.

10.3.1. Lemma. *Let A be an arbitrary $|X|{\times}|Y|$ matrix, and $\mathcal{R}$ a partition of $X{\times}Y$. Then $\|A\|^2 \leq \sum_{R\in\mathcal{R}} \|A_R\|^2$*

Proof: By Fact B.6.3, $\|A\| = \max_{u,v} |u^*Av|$, where the maximum is taken over all unit vectors u, v. Let u, v be unit vectors realizing this maximum. Then we have

$$\|A\| = |u^*Av| = \left| u^* \left(\sum_{R\in\mathcal{R}} A_R \right) v \right| = \left| \sum_{R\in\mathcal{R}} u^* A_R v \right| .$$

Let u^*_R be the portion of u^* corresponding to the rows of R, and v_R be the portion of v corresponding to the columns of R. Notice that $\{u_R\}_{R\in\mathcal{R}}$ do not in general form a partition of u. We now have

$$\begin{aligned} \|A\| &= \left| \sum_{R\in\mathcal{R}} u^*_R A_R v_R \right| \leq \sum_{R\in\mathcal{R}} |u^*_R A_R v_R| \\ &\leq \sum_{R\in\mathcal{R}} \|A_R\| \|u_R\| \|v_R\| \end{aligned}$$

by Fact B.6.3. Applying the Cauchy–Schwarz inequality, we obtain

$$\|A\| \leq \left(\sum_{R\in\mathcal{R}} \|A_R\|^2\right)^{1/2} \left(\sum_{R\in\mathcal{R}} \|u_R\|^2\|v_R\|^2\right)^{1/2}.$$

Now it simply remains to observe that

$$\sum_{R\in\mathcal{R}} \|u_R\|^2\|v_R\|^2 = \sum_{R\in\mathcal{R}}\sum_{(x,y)\in R} u[x]^2v[y]^2 = \|u\|^2\|v\|^2 = 1,$$

as $\mathcal{R}$ is a partition of $X{\times}Y$. □

10.4 Formula Size Lower Bounds

10.4.1 Deterministic Formulae

With this lemma in hand, it is now easy to show that $\mathsf{sumPI}(f)^2$ lower bounds the formula complexity of f. In fact, as this lemma works with respect to an arbitrary rectangle decomposition, we can even show that $\mathsf{sumPI}(f)^2$ lower bounds the rectangle bound, $C^D(R_f)$.

Again, we do not want our final expression to make reference to the structure of monochromatic rectangles in the optimal rectangle partition. Fortunately, in the case of the relation R_f which arises in formula size, there is a very natural covering we can use, together with a monotonicity condition, to get around this. Razborov [Raz90] takes the same path in the rank method, and refers to this covering as the canonical covering. The canonical covering consists of the n sets S_i for $i \in [n]$, where $S_i = \{(x,y) : f(x) = 0, f(y) = 1, x_i \neq y_i\}$. Now any rectangle which is monochromatic with respect to the relation will be monochromatic for some color i and so will be contained in S_i. Razborov's definition of the canonical covering is slightly different than ours, as he takes care to make the elements of the covering rectangles; this is not needed for our application.

10.4.1. Theorem.

$$\mathsf{sumPI}^2(f) \leq C^D(R_f) \leq \mathsf{L}(f)$$

Proof: We have seen that $C^D(R_f) \leq \mathsf{L}(f)$, thus we focus on the inequality $\mathsf{sumPI}^2(f) \leq C^D(R_f)$.

Let $\mathcal{R}$ be a monochromatic rectangle partition of R_f such that $|\mathcal{R}| = C^D(R_f)$, and let A be an arbitrary $|X|{\times}|Y|$ matrix with nonnegative real entries.

By Lemma 10.3.1 we have

$$\|A\|^2 \leq \sum_{S \in \mathcal{R}} \|A_S\|^2 \leq C^D(R_f) \max_{S \in \mathcal{R}} \|A_S\|^2.$$

As each $S \in \mathcal{R}$ is monochromatic, it follows that $S \subseteq S_i$ for some i. It is evident from the definition of spectral norm given in Fact B.6.3 that $\|A_S\| \leq \|A_{S_i}\|$. As S is a rectangle,

$$\begin{aligned} \|A_{S_i}\| = \max_{x,y:\|x\|=\|y\|=1} |x^* A_{S_i} y| &\geq \max_{x_S, y_S:\|x_S\|=\|y_S\|=1} |x_S^* A_{S_i} y_S| \\ &= \|A_S\|, \end{aligned}$$

where x_S, y_S are restricted to have nonzero entries only in the rows and columns of S respectively. This monotonicity condition would hold even if the matrix A were allowed to have negative entries. □

10.4.2 Probabilistic Formulae

The properties of sumPI allow us to show that it can be used to lower bound the probabilistic formula size.

10.4.2. Lemma. *Let $\epsilon < 1/2$. If $f : S \to \{0,1\}$ is ϵ-approximated by functions $\{f_j\}_{j \in J}$ with* $\mathsf{sumPI}(f_j) \leq s$ *for every $j \in J$, then* $\mathsf{sumPI}(f) \leq s/(1-2\epsilon)$.

Proof: By assumption there is a probability distribution $\alpha = \{\alpha_j\}_{j \in J}$ such that $\Pr[f(x) = f_j(x)] \geq 1 - \epsilon$. Thus for a fixed $x \in S$, letting $J_x = \{j \in J : f(x) = f_j(x)\}$, we have $\sum_{j \in J_x} \alpha_j \geq 1 - \epsilon$. Hence for any $x, y \in S$ we have $\sum_{j \in J_x \cap J_y} \alpha_j \geq 1 - 2\epsilon$. For convenience, we write $J_{x,y}$ for $J_x \cap J_y$. As $\mathsf{sumPI}(f_j) \leq s$ there is a family of probability distributions p_j such that whenever $f_j(x) \neq f_j(y)$

$$\sum_{\substack{i \\ x_i \neq y_i}} \sqrt{p_{j,x}(i) p_{j,y}(i)} \geq 1/s.$$

Define $p_x(i) = \sum_{j \in J} \alpha_j p_{j,x}(i)$. Let x, y be such that $f(x) \neq f(y)$.

$$\begin{aligned}
&\sum_{\substack{i \\ x_i \neq y_i}} \sqrt{p_x(i) p_y(i)} \\
&\quad= \sum_{\substack{i \\ x_i \neq y_i}} \sqrt{\sum_{j \in J} \alpha_j p_{j,x}(i)} \sqrt{\sum_{j \in J} \alpha_j p_{j,y}(i))} \\
&\quad\geq \sum_{\substack{i \\ x_i \neq y_i}} \sqrt{\sum_{j \in J_{x,y}} \alpha_j p_{j,x}(i)} \sqrt{\sum_{j \in J_{x,y}} \alpha_j p_{j,y}(i)} \\
&\quad\geq \sum_{\substack{i \\ x_i \neq y_i}} \sum_{j \in J_{x,y}} \sqrt{\alpha_j p_{j,x}(i)} \sqrt{\alpha_j p_{j,y}(i)} \\
&\quad= \sum_{j \in J_{x,y}} \left(\alpha_j \sum_{\substack{i \\ x_i \neq y_i}} \sqrt{p_{j,x}(i) p_{j,y}(i)} \right) \\
&\quad\geq \frac{1 - 2\epsilon}{s},
\end{aligned}$$

where for the third step we have used the Cauchy–Schwarz Inequality. □

This lemma immediately shows that the sumPI method can give lower bounds on probabilistic formula size.

10.4.3. Theorem. *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$. Then*

$$\mathsf{L}^\epsilon(f) \geq ((1 - 2\epsilon)\mathsf{sumPI}(f))^2$$

for any $\epsilon < 1/2$.

Proof: Suppose that $\{f_j\}_{j \in J}$ gives an ϵ-approximation to f. By Lemma 10.4.2 there is some $j \in J$ with $\mathsf{sumPI}(f_j) \geq (1 - 2\epsilon)\mathsf{sumPI}(f)$. Theorem 10.4.1 then implies $\mathsf{L}(f_j) \geq ((1 - 2\epsilon)\mathsf{sumPI}(f))^2$ which gives the statement of the theorem. □

10.5 Beyond the Adversary Method

We have thus far investigated the measure sumPI. This measure is quite natural, and we have seen that it possesses several nice properties. If we are willing to do more work in finding out the structure of the monochromatic rectangles of the relation R_f, then we can apply Lemma 10.3.1 in a stronger way to obtain larger formula size bounds. We present one such approach here, which we call maxPI.

We feel obliged to remark at the outset that while sumPI can be computed in time polynomial in the size of the truth table of f by reduction to semidefinite programming, we do not know how to compute maxPI in less than exponential time.

We define maxPI in its primal form:

10.5.1. Definition.

$$\mathsf{maxPI}(f) = \min_{p} \max_{\substack{x,y \\ f(x)\neq f(y)}} \frac{1}{\max_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)}}.$$

Notice that the difference between maxPI and sumPI is that here we take the maximum over the indices instead of the sum. This is why we call this technique "maximum probability of indices". As the maximum over the indices will be less than the sum over indices, it is obvious that $\mathsf{sumPI}(f) \leq \mathsf{maxPI}(f)$. In the remainder of this section we will show that $(\mathsf{maxPI}(f))^2$ also lower bounds $C^D(R_f)$. Later we give an example of a partial function which shows that $\mathsf{maxPI}(f)$ is not in general a lower bound on the quantum query complexity of f and we show that for total functions $\mathsf{sumPI}(f)$ and $\mathsf{maxPI}(f)$ are polynomially related.

To show that $(\mathsf{maxPI}(f))^2$ is a lower bound on $C^D(R_f)$, we first partially dualize the expression maxPI. The final expression remains a minimization problem, but we minimize over discrete index selection functions, instead of families of probability distributions, which makes it much more tractable.

10.5.2. Definition (Index selection functions). *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, $X{=}f^{-1}(0)$, and $Y{=}f^{-1}(1)$. For $i \in [n]$ let D_i be an $|X|{\times}|Y|$ matrix defined by $D_i[x,y] = 1 - \delta_{x_i,y_i}$. We call the set of n Boolean ($0-1$) matrices $\{P_i\}_{i\in n}$* index selection functions *if*

1. *$\sum_i P_i = E$, where $E[x,y] = 1$ for every $x \in X$, $y \in Y$. (informally: for every $x \in X$, $y \in Y$ we select a unique index)*
2. *$P_i \leq D_i$ (informally: for every $x \in X$, $y \in Y$ the index we select is an i such that $x_i \neq y_i$).*

Notice that index selection functions correspond to partitioning $X{\times}Y$, in such a way that if x, y are in the ith part, then $x_i \neq y_i$.

10.5.3. Theorem (Spectral adversary version of maxPI). *Let f, X, Y be as in the previous definition. Let A be an arbitrary $|X|{\times}|Y|$ nonnegative matrix satisfying $A[x,y] = 0$ whenever $f(x) = f(y)$. Then*

$$\mathsf{maxPI}(f) = \min_{\{P_i\}_i} \max_{A} \frac{\|A\|}{\max_i \|A \circ P_i\|},$$

where $\{P_i\}_i$ runs through all index selection functions.

Proof: For a fixed family of probability distributions $p = \{p_x\}$, and for the expression

$$\max_{\substack{x,y \\ f(x)\neq f(y)}} \frac{1}{\max_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)}}, \tag{10.3}$$

let us define the index selection function $P_i[x,y] = 1$ if $i = \operatorname{argmax}_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)}$ and 0 otherwise. (Argmax is the smallest argument for which the expression attains its maximal value.) Then the denominator in Equation (10.3) becomes equal to $\sum_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)} P_i[x,y]$. If we replace the above system of P_is with any other choice of index selection function the value of $\sum_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)} P_i[x,y]$ will not increase. Thus we can rewrite Equation (10.3) as

$$\max_{\substack{x,y \\ f(x)\neq f(y)}} \frac{1}{\max_{\{P_i\}_i} \sum_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)} P_i[x,y]},$$

where here $P_i[x,y]$ runs through all index selection functions. Thus:

$$\mathsf{maxPI}(f) =$$

$$1 \Big/ \max_p \min_{\substack{x,y \\ f(x)\neq f(y)}} \max_{\{P_i\}_i} \sum_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)} P_i[x,y]. \tag{10.4}$$

Notice that in Equation (10.4) the minimum is interchangeable with the second maximum. The reason for this is that for a fixed p there is a fixed $\{P_i[x,y]\}_i$ system that maximizes $\sum_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)} P_i[x,y]$ for all $x, y : \ f(x) \neq f(y)$. Thus:

$$\mathsf{maxPI}(f) =$$

$$1 \Big/ \max_{\{P_i\}_i} \max_p \min_{\substack{x,y \\ f(x)\neq f(y)}} \sum_{i:x_i\neq y_i} \sqrt{p_x(i)p_y(i)} P_i[x,y].$$

Following the proof of the main theorem of Špalek and Szegedy [ŠS05] we can create the semidefinite version of the above expression. The difference here, however, is that we have to treat $\{P_i\}_i$ (the index selection functions) as a "parameter" of the semidefinite system over which we have to maximize. Unfortunately it also appears in the final expression.

Semidefinite version of maxPI: For fixed $\{P_i\}_i$ let $\mu'_{\max}$ be the solution of the following semidefinite program:

$$\begin{aligned} &\text{maximize } \mu' \\ &\text{subject to} \quad (\forall i) \quad R_i \succeq 0, \\ &\qquad\qquad \textstyle\sum_i R_i \circ I = I, \\ &\qquad\qquad \textstyle\sum_i R_i \circ P_i \geq \mu' F. \end{aligned}$$

Define $\mu_{\max}$ as the maximum of $\mu'_{\max}$, where P_i $(1 \leq i \leq n)$ run through all index selection functions. Then $\mathsf{maxPI} = 1/\mu_{\max}$.

To see how to create a solution to the semidefinite program from probability distributions $\{p_x, p_y\}$: let $R_i[x,y] = \sqrt{p_x(i)p_y(i)}$. Then $\sum R_i \circ I = I$ follows as the $\{p_x\}$ are probability distributions. Furthermore, the $\{R_i\}$ are positive semidefinite as they are rank one matrices with nonnegative entries. Finally, if $f(x) \neq f(y)$ then

$$\sum_i R_i \circ P_i = \sum_{i:x_i \neq y_i} \sqrt{p_x(i)p_y(i)} P_i[x,y].$$

For the other direction, let $\{R_i\}$ be a solution to the semidefinite program which realize the bound $1/\mu$. We show how to change this into a rank one solution whose bound is as good. As each R_i is positive semidefinite, we can take a Cholesky decomposition where $R_i = X_i X_i^T$. Define a column vector $q_i[x] = \sqrt{\sum_j X_i[x,j]^2}$. Let R_i' be the rank one matrix defined as $R_i' = q_i q_i^T$. That $\sum_i R_i' = I$ follows from the same property of $\{R_i\}$. By the Cauchy-Schwarz inequality,

$$R_i[x,y] = \sum_j X[x,j]X[j,y] \leq \sqrt{\sum_j X[x,j]^2}\sqrt{\sum_j X[j,y]^2} = q_i[x]q_i[y] = R_i'[x,y].$$

This implies that $\sum_i R_i'[x,y]P_i[x,y] \geq R_i[x,y]P_i[x,y] \geq \mu$ and so thus the bound of the original definition of maxPI is less than that given by the semidefinite formulation.

To conclude the proof, we can now dualize the semidefinite version of maxPI and simplify it in same way as was done in Špalek and Szegedy for the case of sumPI. The only change is that D_i needs to be replaced with P_i, and we have to minimize over all choices of $\{P_i\}_i$. □

10.5.4. Theorem.

$$\mathsf{sumPI}^2(f) \leq \mathsf{maxPI}^2(f) \leq C^D(R_f) \leq \mathsf{L}(f)$$

Proof: The only inequality we have not seen is $\mathsf{maxPI}^2(f) \leq C^D(R_f)$.

Let $\mathcal{R}$ be a monochromatic rectangle partition of R_f such that $|\mathcal{R}| = C^D(R_f)$, and let A be an arbitrary $|X| \times |Y|$ matrix with nonnegative real entries. For $S \in \mathcal{R}$ let color(S) be the least index c such that $x_c \neq y_c$ holds for all $(x,y) \in S$. By assumption each S is monochromatic, thus such a color exists. Define

$$S_c = \cup_{\text{ color } (R)=c} R.$$

Then it is clear that if $S \in \mathcal{R}$ then $S \subseteq S_c$ where $c = \text{color}(S)$. By the same argument as in Theorem 10.4.1 we have $\|A_S\| \leq \|A_{S_c}\|$, and so

$$\|A\|^2 \leq \sum_{S \in \mathcal{R}} \|A_S\|^2 \leq C^D(R_f) \max_{c \in [n]} \|S_c\|^2$$

It thus follows that

$$\max_{A \neq 0} \frac{\|A\|^2}{\max_{c \in [n]} \|A_{S_c}\|^2} \leq C^D(R_f).$$

We have exhibited a particular index selection function, the $\{S_c\}_c$, for which this inequality holds, thus it also holds for $\mathsf{maxPI}^2(f)$ which is the minimum over all index selection functions. □

10.6 Comparison with Previous Methods

In this section we compare some of the formula size methods discussed in Chapter 8 with our methods. We see that the sumPI^2 measure generalizes all of the Hamming distance 1 methods discussed there.

10.6.1 Khrapchenko's method

We see that Khrapchenko's method is a special case of the probability scheme.

10.6.1. Theorem. *Let $S \subseteq \{0,1\}^n$ and $f : S \to \{0,1\}$. Let $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$. Let C be the set of pairs $(x,y) \in A{\times}B$ with Hamming distance 1, that is $C = \{(x,y) \in A{\times}B : d_H(x,y) = 1\}$. Then $\mathsf{sumPI}(f)^2 \geq \frac{|C|^2}{|A||B|}$.*

Letting A, B, C be as in the statement of the theorem, we set up our probability distributions as follows:

- $p_A(x){=}1/|A|$ for all $x{\in}A$, $p_A(x){=}0$ otherwise
- $p_B(x){=}1/|B|$ for all $x{\in}B$, $p_B(x){=}0$ otherwise
- $q(x,y){=}1/|C|$ for all $(x,y){\in}C$, $q(x,y){=}0$ otherwise
- $p_{x,i}(y){=}1$ if $(x,y){\in}C$ and $x_i \neq y_i$, 0 otherwise. Note that this is a probability distribution as for every x there is only one y such that $(x,y){\in}C$ and $x_i \neq y_i$.

10.6.2 The Koutsoupias bound

The Koutsoupias bound is a Hamming distance 1 version of the spectral version of sumPI^2.

10.6.2. Theorem. *Let $f : \{0,1\}^n \to \{0,1\}$, and let $A \subseteq f^{-1}(0)$, and $B \subseteq f^{-1}(1)$. Let $C = \{(x,y) \in A{\times}B : d_H(x,y) = 1\}$. Let Q be a $|A|{\times}|B|$ matrix $Q[x,y] = C(x,y)$ where C is identified with its characteristic function. Then $\mathsf{sumPI}(f)^2 \geq \|Q\|^2$.*

Proof: The bound follows easily from the the spectral version of sumPI. Let Q be as in the statement of the theorem. Notice that since we only consider pairs with Hamming distance 1, for every row and column of Q_i there is at most one nonzero entry, which is at most 1. Thus by Fact B.6.4 we have $\|Q_i\|^2 \leq \|Q\|_1 \|Q\|_\infty \leq 1$. □

10.6.3 Håstad's method

10.6.3. Theorem. *Let $f : \{0,1\}^n \to \{0,1\}$. Let A be the event that a random restriction in R_p reduces f to the constant 0, B be the event that a random restriction in R_p reduces f to the constant 1, and let C be the event that a random restriction $\rho \in R_p$ is such that $f|_\rho$ is a single literal. Then*

$$\mathsf{sumPI}(f)^2 \geq \frac{Pr[C|\Delta]^2}{Pr[A|\Delta]Pr[B|\Delta]} \left(\frac{1-p}{2p}\right)^2$$

Proof: We see that Håstad's lemma follows from the probability scheme. In this proof we only consider restrictions obtained from R_p that are in the filter Δ. We also abuse notation and use A and B to mean the sets of restrictions in Δ which contribute with non-zero probability to the events A and B respectively.

Implicit in Håstad's proof is the following relation between restrictions in A and B. For every $\rho \in C$, $f|_\rho$ reduces to a single literal, that is, for every $\rho \in C$, there is an i such that $f|_\rho = x_i$ (or $\neg x_i$ if the variable is negated). Define ρ^b to be ρ where x_i is set to b, for $b \in \{0,1\}$ (set x_i to $1-b$ if the variable is negated). To fit into the framework of the probability scheme, let $\overline{\rho^b}$ be ρ^b where all remaining $\star$s are set to 1. This doesn't change the value of the function, because it is already constant on ρ^b. Then we say that $\overline{\rho^0}, \overline{\rho^1}$ are in the relation.

We set $p_A(\sigma) = \frac{\Pr[\sigma]}{\Pr[A|\Delta]}$ for any $\sigma \in A$, and $p_B(\tau) = \frac{\Pr[\tau]}{\Pr[B|\Delta]}$ for any $\tau \in B$, and for every pair $\overline{\rho^0}, \overline{\rho^1}$ in the relation, where $\rho \in C$, $f|_\rho = x_i$ or $\neg x_i$, let

$$\begin{aligned} p'_{\overline{\rho^0},i}(\overline{\rho^1}) &= 1 \\ p'_{\overline{\rho^1},i}(\overline{\rho^0}) &= 1 \\ q(\overline{\rho^0}, \overline{\rho^1}) &= \frac{Pr[\rho]}{Pr[C|\Delta]} \end{aligned}$$

The probabilities are 0 on all other inputs. We can easily verify that the probabilities sum to 1. For p', notice that the Hamming distance between $\overline{\rho^0}$ and $\overline{\rho^1}$ is 1, so when $\overline{\rho^b}$ and i are fixed, there is only a single $\overline{\rho^{1-b}}$ with probability 1.

By Theorem 10.2.4 and Theorem 10.4.1,

$$\begin{aligned} \mathsf{L}(f) &\geq \frac{p_A(x)p_B(y)p'_{y,i}(x)p'_{x,i}(y)}{q(x,y)^2} \\ &= \frac{Pr[\rho^0]}{Pr[A|\Delta]}\frac{Pr[\rho^1]}{Pr[B|\Delta]}\left(\frac{Pr[C|\Delta]}{Pr[\rho]}\right)^2 \end{aligned}$$

Finally, notice that $Pr[\rho] = \frac{2p}{1-p}Pr[\rho^b]$. □

Remark Håstad actually defines $f|_\rho$ to be the result of reducing the formula for f (not the function) by applying a sequence of reduction rules, for each restricted variable. So there is a subtlety here about whether $f|_\rho$ denotes the reduced formula, or the reduced function, and the probabilities might be different if we are in one setting or the other. However both in his proof and ours, the only thing that is used about the reduction is that if the formula or function reduces to a single literal, then fixing this literal to 0 or to 1 reduces the function to a constant. Therefore, both proofs go through for both settings.

10.7 Limitations

10.7.1 Hamming distance 1 techniques

We show that the bounds for a function f given by Khrapchenko's and Koutsoupias' method, and by Håstad's lemma are upper bounded by the product of the zero sensitivity and the one sensitivity of f. We will later use this bound to show a function on n bits for which the best lower bound given by these methods is n and for which an $\approx n^{1.32}$ bound is provable by sumPI^2.

10.7.1. Lemma. *The bound given by the Khrapchenko method (Theorem 10.6.1), Koutsoupias' method (Theorem 10.6.2), and Håstad's Lemma (Theorem 10.6.3) for a function f are at most $s_0(f)s_1(f) \leq s^2(f)$.*

Proof: Let A be a nonnegative matrix, with nonzero entries only in positions (x, y) where $f(x) = 0, f(y) = 1$ and the Hamming distance between x, y is one. We first show that

$$\max_A \frac{\|A\|^2}{\max_i \|A_i\|^2} \leq s_0(f)s_1(f). \tag{10.5}$$

Let a_{max} be the largest entry in A. A can have at most $s_0(f)$ many nonzero entries in any row, and at most $s_1(f)$ many nonzero entries in any column, thus by Fact B.6.4,

$$\|A\|^2 \leq \|A\|_1\|A\|_\infty \leq a_{max}^2 s_0(f)s_1(f).$$

On the other hand, for some i, the entry a_{max} appears in A_i, and so by Fact B.6.3, $\|A_i\|^2 \geq a_{max}^2$. Equation (10.5) follows.

Now we see that the left hand side of Equation (10.5) is larger than the three methods in the statement of the theorem. That it is more general than Koutsoupias method is clear. To see that it is more general than the probability schemes method where $q(x, y)$ is only positive if the Hamming distance between x, y is one: given the probability distributions q, p_X, p_Y, define the matrix $A[x, y] = q(x, y)/\sqrt{p_X(x)p_Y(y)}$. By Fact B.6.3, $\|A\| \geq 1$, witnessed by the unit vectors $u[x] = \sqrt{p_X(x)}$ and $v[y] = \sqrt{p_Y(y)}$. As each reduced matrix A_i has at most one nonzero entry in each row and column, by Fact B.6.4 we have

$$\max_i \|A_i\|^2 \leq \max_{x,y} \frac{q^2(x, y)}{p_X(x)p_Y(y)}.$$

Thus we have shown

$$\max_A \frac{\|A\|^2}{\max_i \|A_i\|^2} \geq \max_{p_X, p_Y, q} \min_{x,y} \frac{p_X(x)p_Y(y)}{q^2(x, y)}.$$

□

The only reference to the limitations of these methods we are aware of is Schürfeld [Sch83], who shows that Khrapchenko's method cannot prove bounds greater than $C_0(f)C_1(f)$.

10.7.2 Limitations of sumPI and maxPI

The limitations of the adversary method are well known [Amb02, LM04, Sze03, Zha05, ŠS05]. Špalek and Szegedy, in unifying the adversary methods, also give the most elegant proof of their collective limitation. The same proof also shows the same limitations hold for the maxPI measure.

10.7.2. Lemma. *Let $f : \{0, 1\}^n \to \{0, 1\}$ be any partial or total Boolean function. If f is total (respectively, partial) then* $\mathsf{maxPI}(f) \leq \sqrt{C_0(f)C_1(f)}$ *(respectively,* $\min\{\sqrt{nC_0(f)}, \sqrt{nC_1(f)}\}$*).*

Proof: Assume that f is total. Take x, y such that $f(x) = 0$ and $f(y) = 1$. We choose any 0-certificate B_0 for x with $|B_0| = C_0(f)$ and any 1-certificate B_1 for y with $|B_1| = C_1(f)$ and let $p_x(i) = 1/C_0(f)$ for all $i \in B_0$ and $p_y(i) = 1/C_1(f)$ for all $i \in B_1$. As f is total, we have $B_0 \cap B_1 \neq \emptyset$, thus let $j \in B_0 \cap B_1$. For this j we have $p_x(j)p_y(j) \geq 1/(C_0(f)C_1(f))$, thus $\min_i 1/p_x(i)p_y(i) \leq C_0(f)C_1(f)$.

The case where f is partial follows similarly. As we no longer know that $B_0 \cap B_1 \neq \emptyset$, we put a uniform distribution over a 0-certificate of x and the uniform distribution over $[n]$ on y or vice versa. □

This lemma implies that sumPI and maxPI are polynomially related for total f.

10.7.3. Corollary. *Let f be a total Boolean function. Then*

$$\mathsf{maxPI}(f) \leq \mathsf{sumPI}^4(f).$$

Proof: By [Amb02, Thm. 5.2] we know that $\sqrt{bs(f)} \leq \mathsf{sumPI}(f)$. As f is total, by the above lemma we know that $\mathsf{maxPI}(f) \leq \sqrt{C_0(f)C_1(f)}$. This in turn is smaller than $bs(f)^2$ as $C(f) \leq s(f)bs(f)$ [Nis91]. The statement follows. □

Besides the certificate complexity barrier, another serious limitation of the sumPI method occurs for partial functions where every positive input is far in Hamming distance from every negative input. Thus for example, if for any pair x, y where $f(x) = 1$ and $f(y) = 0$ we have $d_H(x, y) \geq \epsilon n$, then by putting the uniform distribution over all input bits it follows that $\mathsf{sumPI}(f) \leq 1/\epsilon$. The measure maxPI does not face this limitation as there we still only have one term in the denominator.

Following this line of thinking, we can give an example of a partial function f where $\mathsf{maxPI}(f) \gg \mathsf{sumPI}(f)$. Such an example is the Collision problem (see Section 10.8.3), as here any positive and negative input must differ on at least $n/2$ positions. Another family of examples comes from property testing, where the promise is that the input either has some property, or that it is ϵ-far from having the property.

10.8 Concrete lower bounds

The quantum adversary argument has been used to prove lower bounds for a variety of problems. Naturally, all of these lower bounds carry over to formula size lower bounds. In this section we present some new lower bounds, in order to highlight the strengths and weaknesses of maxPI and sumPI.

10.8.1 Recursive majorities

As an example of applying sumPI, we look at the recursive majority of three function. We let $\mathsf{R{-}MAJ}_3^h : \{0,1\}^{3^h} \to \{0,1\}$ be the function computed by a complete ternary tree of depth h where every internal node is labeled by a majority gate and the input is given at the leaves.

Recursive majority of three has been studied before in various contexts. It is a monotone function which is very sensitive to noise [MO03], making it useful for hardness amplification in NP [O'D02]. Jayram, Kumar, and Sivakumar [JKS03] give nontrivial lower and upper bounds on the randomized decision tree complexity of recursive majority of three. They show a lower bound of $(7/3)^h$ on the randomized decision tree complexity. As far as we know, the quantum query complexity of recursive majority of three has not yet been investigated. We show a lower bound of 2^h on the quantum query complexity.

10.8.1. Lemma. $\mathsf{sumPI}(\mathsf{R{-}MAJ}_3^h) = \mathsf{maxPI}(\mathsf{R{-}MAJ}_3^h) = 2^h$

Proof: To see that $\mathsf{maxPI}(\mathsf{R{-}MAJ}_3^h) \leq 2^h$, observe that

$$C_0(\mathsf{R{-}MAJ}_3^h) = C_1(\mathsf{R{-}MAJ}_3^h) = 2^h.$$

The result then follows from Lemma 10.7.2.

We now turn to the lower bound. We will first show a lower bound for $\mathsf{R{-}MAJ}_3^1$, the majority of three function, and then apply Lemma 10.2.7. Consider the following table, where the rows are indexed by negative instances x, the columns by positive instances y, and 1's indicate when $d_H(x, y) = 1$.

	110	101	011
001	0	1	1
010	1	0	1
100	1	1	0

Interpreting this table as the adjacency matrix of a graph, it is clear that every vertex has degree 2. Thus Khrapchenko's method gives a bound of 4 for the base function. By Theorem 10.6.1 we have $\mathsf{sumPI}(\mathsf{R{-}MAJ}_3^1) \geq 2$. Now applying Lemma 10.2.7 gives the lemma. □

From Lemma 10.8.1 we immediately obtain quantum query complexity and formula size lower bounds:

10.8.2. Theorem. *Let $\mathsf{R{-}MAJ}_3^h$ be the recursive majority of three function of height h. Then $Q_\epsilon(\mathsf{R{-}MAJ}_3^h) \geq (1-2\sqrt{\epsilon(1-\epsilon)})2^h$ and $\mathsf{L}^\epsilon(\mathsf{R{-}MAJ}_3^h) \geq (1-2\epsilon)4^h$.*

The best upper bound on the formula size of $\mathsf{R{-}MAJ}_3^h$ is 5^h. For this bound, we will use the following simple proposition about the formula size of iterated functions.

10.8.3. Proposition. *Let $S \subseteq \{0,1\}^n$ and $f : S \rightarrow \{0,1\}$. If $\mathsf{L}(f) \leq s$ then $\mathsf{L}(f^d) \leq s^d$, where f^d is the d^{th} iteration of f.*

10.8.4. Proposition. $\mathsf{L}(\mathsf{R{-}MAJ}_3^h) \leq 5^h$.

Proof: The formula $(x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$ computes $\mathsf{R{-}MAJ}_3^1$ and has 5 leaves. Using Proposition 10.8.3 gives $\mathsf{L}(\mathsf{R{-}MAJ}_3^h) \leq 5^h$. □

10.8.2 Ambainis' function

We define a function $\mathrm{f_A} : \{0,1\}^4 \rightarrow \{0,1\}$ after Ambainis [Amb03]. This function evaluates to 1 on the following values: 0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000. That is, $f(x) = 1$ when $x_1 \leq x_2 \leq x_3 \leq x_4$ or $x_1 \geq x_2 \geq x_3 \geq x_4$. To obtain this formulation from Ambainis' original definition, exchange x_1 and x_3, and take the negation of the resulting function. There are a few things to notice about this function. The sensitivity of $\mathrm{f_A}$ is 2 on every input. Also on an input $x = x_1x_2x_3x_4$ the value of $\mathrm{f_A}(x)$ changes if both bits sensitive to x are flipped simultaneously, and if both bits insensitive for x are flipped simultaneously.

We will be looking at iterations of the base function $\mathrm{f_A}$ as in Definition 10.2.6. Notice that the sensitivity of ${\mathrm{f_A}}^d$ is 2^d on every input $x \in \{0,1\}^{4^d}$.

10.8.5. Lemma. *$\mathsf{sumPI}({\mathrm{f_A}}^d) = 2.5^d$.*

Proof: Ambainis has already shown that $\mathsf{sumPI}({\mathrm{f_A}}^d) \geq 2.5^d$ [Amb03].

We now show the upper bound. We will show an upper bound for the base function $\mathrm{f_A}$ and then use the composition Lemma 10.2.5. Every input $x_1x_2x_3x_4$ has two sensitive variables and two insensitive variables. For any $x \in \{0,1\}^4$ we set $p_x(i) = 2/5$ if i is sensitive for x and $p_x(i) = 1/10$ if i is insensitive for x. The claim follows from the following observation: for any $x, y \in \{0,1\}^4$ such that $f(x) \neq f(y)$ at least one of the following holds

- x and y differ on a position i which is sensitive for both x and y. Thus $\sum_i \sqrt{p_x(i)p_y(i)} \geq 2/5$
- x and y differ on at least 2 positions, each of these positions being sensitive for at least one of x, y. Thus $\sum_i \sqrt{p_x(i)p_y(i)} \geq 2\sqrt{1/25} = 2/5$

□

This lemma gives us a bound of $6.25^d \approx N^{1.32}$ on the formula size of ${\mathrm{f_A}}^d$. Since the sensitivity of ${\mathrm{f_A}}^d$ is 2^d, by Lemma 10.7.1, the best bound provable by Khrapchenko's method, Koutsoupias' method, and Håstad's lemma is $4^d = N$.

It is natural to ask how tight this formula size bound is. The best upper bound we can show on the formula size of ${\mathrm{f_A}}^d$ is 10^d.

10.8.6. Proposition. *$\mathsf{L}({\mathrm{f_A}}^d) \leq 10^d$*

Proof: It can be easily verified that the following formula of size 10 computes the base function $\mathrm{f_A}$:

$$(\neg x_1 \vee x_3 \vee \neg x_4) \wedge ((\neg x_1 \wedge x_3 \wedge x_4) \vee ((x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)))\,. \tag{10.6}$$

This formula was found by computer search. The claim now follows from Proposition 10.8.3. □

10.8.3 Collision problem

In this section we look at the collision function, $\mathrm{f_C}$. This is a partial function, where for an alphabet Σ the defined inputs $x = x_1 x_2 \dots x_n \in \Sigma^n$ satisfy one of the following conditions:

- All x_i are different
- For each i there exists exactly one $j \neq i$ such that $x_i = x_j$.

Those inputs satisfying the first condition are positive inputs and those satisfying the second condition are negative. An optimal lower bound for the quantum query complexity of $\Omega(n^{1/3})$ has been given by Aaronson and Shi [AS04]. We now show that the quantum adversary method cannot give better than a constant bound for this problem.

10.8.7. Lemma. $\mathsf{sumPI}(\mathrm{f_C}) \leq 2$

Proof: We demonstrate a set of probability distributions p_x such that for any positive instance x and negative instance y we have

$$\sum_{\substack{i \\ x_i \neq y_i}} \sqrt{p_x(i) p_y(i)} \geq 1/2.$$

The upper bound then follows.

Our probability distribution is very simple: for every x, let $p_x(i)$ be the uniform distribution over $[n]$. Any positive and negative instance must disagree in at least $n/2$ positions, thus

$$\sum_{\substack{i \\ x_i \neq y_i}} \sqrt{p_x(i) p_y(i)} \geq \frac{n}{2} \sqrt{\frac{1}{n} \frac{1}{n}} = \frac{1}{2}.$$

□

On the other hand, $\mathsf{maxPI}(\mathrm{f_C}) \geq \sqrt{n/2}$. As there is an upper bound for the collision problem of $O(n^{1/3})$ by Brassard, Høyer, Tapp [BHT97], this also shows that in general $\mathsf{maxPI}(f)$ is not a lower bound on the quantum query complexity of f.

10.8.8. Lemma. $\mathsf{maxPI}(\mathrm{f_C}) = \Theta(\sqrt{n})$

Proof: For the upper bound: On every positive instance x, where all x_i are different, we put the uniform distribution over $i \in [n]$; for a negative instance y we put probability $1/2$ on the first position, and probability $1/2$ on the position j such that $y_1 = y_j$. As $y_1 = y_j$, any positive instance x must differ from y

on position 1 or position j (or both). Thus $\max_{i,x_i \neq y_i} p_x(i)p_y(i) \geq 1/2n$ and $\mathsf{maxPI}(\mathrm{f_C}) \leq \sqrt{2n}$.

Now for the lower bound. Fix a set of probability distributions p_x. Let x be any positive instance. There must be at least $n/2$ positions i satisfying $p_x(i) \leq 2/n$. Call this set of positions I. Now consider a negative instance y of where $y_j = x_j$ for all $j \notin I$, and y is assigned values in I in an arbitrary way so as to make it a negative instance. For this pair x, y we have $\max_i \sqrt{p_x(i)p_y(i)} \leq \sqrt{2/n}$, thus $\mathsf{maxPI}(\mathrm{f_C}) \geq \sqrt{n/2}$. □

The following table summarizes the bounds from this section.

Function	Input size	sum PI	Q_ϵ	max PI	L	$s_0 s_1$
$\mathsf{R{-}MAJ}_3^h$	$N = 3^h$	$2^h \approx N^{0.63}$	$\Omega(N^{0.63})$	$N^{0.63}$	$\Omega(N^{1.26})$, $O(N^{1.46})$	$N^{1.26}$
${\mathrm{f_A}}^h$	$N = 4^h$	$2.5^h \approx N^{0.66}$	$\Omega(N^{0.66})$ [Amb03]	$\leq 3^h \approx N^{0.69}$	$\Omega(N^{1.32})$, $O(N^{1.66})$	N
$\mathrm{f_C}$	N	2	$\Theta(N^{1/3})$	$\Theta(\sqrt{N})$		

10.9 Conclusions

An outstanding open problem is whether the square of the quantum query complexity lower bounds the formula size. We have given some support to this conjecture by showing it is true for one of the two main techniques of proving lower bounds on quantum query complexity. It would also be interesting to investigate if the same is true of approximate polynomial degree, the other main lower bound technique for quantum query complexity.

We have seen that while in the spectral version of the adversary method the maximum is restricted to nonnegative matrices, its square remains a bound on formula size when the maximum is taken over arbitrary matrices. Can we take advantage of this to prove larger lower bounds? In particular, the equivalence between the spectral version and the min max formulation appears to break down when the spectral version is quantified over arbitrary matrices. Thus we can also ask: do the same certificate complexity limitations still apply to the spectral adversary method taking the maximum over arbitrary matrices?

Finally, notice that for the case of the spectral norm we were able to prove a stronger lemma, with respect to arbitrary rectangle partitions, than we were able to do in Chapter 9. We would conjecture, however, that this case should work as well with respect to arbitrary rectangle decompositions. More formally, we conjecture the following:

10.9.1. Conjecture. *Let A be a matrix over $X \times Y$ with $n = \min\{|X|, |Y|\}$*

and let $\mathcal{R}$ be a rectangle partition of $X \times Y$. Then for any $1 \leq k \leq n$

$$\sum_{i=1}^{k} \sigma_i^2(A) \leq \sum_{R \in \mathcal{R}} \sum_{i=1}^{k} \sigma_i^2(A_R)$$

It would also be quite interesting if this conjecture were not the case because then we would have a method which lower bounds communication complexity, but not the rectangle bound. As we have seen, practically all of the generic techniques available actually lower bound the rectangle bound, and there are very few examples known of separations between these two measures. Currently, the largest separation between communication complexity and the log of the rectangle bound is a square, shown by [KLO96].

Appendix A

Approximate Lower Bound Counting with Arthur–Merlin Games

A.1 Approximate lower bound counting

A.1.1. Definition. We call $A \subseteq \{0,1\}^* \times \{0,1\}^*$ an *NP-relation* if

- A is polynomial time decidable
- There is a polynomial $p(\cdot)$ such that $\ell(y) \leq p(\ell(x))$ for every $(x,y) \in A$.

We will let $A_x = \{y : (x,y) \in A\}$.

A.1.2. Theorem (Babai). *Let A be an NP-relation. There is an* AM *algorithm* LBcount *such that*

- *If* $|A_x| \geq 2^{k+1}$ *then* $\Pr_r[\text{LBcount}(x,r) = 1] = 1$
- *If* $|A_x| \leq 2^{k-1}$ *then* $\Pr_r[\text{LBcount}(x,r) = 1] \leq 1/3$

Proof: Fix $x \in \{0,1\}^n$ and say that $A_x \subseteq \{0,1\}^{n'}$. Notice that by considering the direct product $A_x \times \ldots \times A_x$ of A_x with itself d times, we can amplify the gap between the two cases we have to distinguish from 4 to 4^d. We will do this for $d = \log n' + 1$. As n' is polynomial in n, the resulting direct product remains an NP relation. Thus without loss of generality we can assume we have a NP relation B with $B_x \subseteq \{0,1\}^m$ and we must show

- If $|B_x| \geq 2^{k+2\log m}$ then $\Pr_r[\text{LBcount}(x,r) = 1] = 1$
- If $|A_x| \leq 2^{k-1}$ then $\Pr_r[\text{LBcount}(x,r) = 1] \leq 1/3$

The Arthur–Merlin algorithm is as follows:

- Arthur chooses at random $2m$ many k-by-m matrices $M_1, \dots, M_{2m}$ with entries from $\{0,1\}$. Notice that m is polynomial in the input length n as B is an NP-relation.
- Merlin returns $2m+1$ strings to Arthur, labelled y and y_i for each $1 \leq i \leq 2m$.
- If $y, y_i \in B_x$ and $M_i y = M_i y_i$ for each $1 \leq i \leq 2m$ then Arthur accepts. Otherwise, he rejects.

We will prove the completeness and the soundness of the algorithm separately in the next two claims:

A.1.3. Claim (Completeness). *If $|B_x| \geq 2^{k+2\log m}$ then $\exists y, y_1, \dots, y_{2m} \in B_x$ with $M_i y = M_i y_i$ for all $i \in [2m]$ and* any *set of k-by-n' matrices $\{M_i\}$.*

Proof: We argue the first item by Kolmogorov complexity. Fix $M_1, \dots, M_{2m}$. By the first pillar of Kolmogorov complexity, there is an element $y \in B_x$ with $C(y \mid B_x, M_1, \dots, M_{2m}) \geq k + 2\log m$. Suppose for contradiction that for some i there is no y_i in B_x with $M_i y = M_i y_i$. Then given $M_1, \dots, M_{2m}$ we can describe y by giving the index i, the string $z = M_i y$, and a constant number of bits to say "y is the unique solution to $M_i v = z$ in A_x". This description is of length $k + \log m + O(1)$, a contradiction. □

A.1.4. Claim (Soundness). *If $|B_x| \leq 2^{k-1}$ then*

$$\Pr_{M_1,\dots,M_{2m}}[\exists y, y_1, \dots, y_{2m} : M_1 y = M_1 y_1 \wedge \dots \wedge M_{2m} y = M_{2m} y_{2m}] \leq 2^{-m}.$$

Proof: The second item can be argued as follows: fix y and $y' \neq y$. Notice that the probability that the i^{th} entry of My and My' agree under a random matrix M is $1/2$. Thus by independence of the entries of M we have $\Pr_M[My = My'] = 2^{-m}$. It then follows by a union bound that

$$\Pr_M[\exists y' : My = My'] \leq 2^{k-m} \leq 1/2.$$

As Arthur chooses the M_i independently, we have

$$\Pr_{M_1,\dots,M_{2m}}[\exists y_1, \dots, y_{2m} : M_1 y = M_1 y_1 \wedge \dots \wedge M_{2m} y = M_{2m} y_{2m}] \leq 2^{-2m}$$

Now again by a union bound we have

$$\Pr_{M_1,\dots,M_{2m}}[\exists y, y_1, \dots, y_{2m} : My = My_1 \wedge \dots \wedge My = My_{2m}] \leq 2^{-m}.$$

□

□

Appendix B

Linear Algebra

B.1 Terminology

In this thesis we will always work over the field of real numbers $\mathbb{R}$ or complex numbers $\mathbb{C}$. For definiteness in this appendix, we will work over $\mathbb{C}$—the modifications needed for $\mathbb{R}$ are generally straightforward.

We consider a $m \times n$ matrix A as a m-by-n array of scalars from $\mathbb{C}$. We think of a vector of dimension m as a m-by-1 matrix. Let $A[i, j]$ denote the (i, j) entry of A. For a vector v we will usually use the shorthand $v[i] = v[i, 1]$. A special matrix is the *identity matrix* of size n, written I_n, which has $I_n[i, i] = 1$ for $1 \leq i \leq n$ and all other entries equal to zero. When the dimension is clear from the context we will omit the subscript.

The *transpose* of A, written A^T is defined as $A^T[i, j] = A[j, i]$. We will also make use of the *conjugate transpose* of A, written A^*. Remember that for a complex number $z = a + bi \in \mathbb{C}$ the complex conjugate of z is $\overline{z} = a - bi$. Thus $A^*[i, j] = \overline{A[j, i]}$. A matrix is called *Hermitian* if $A = A^*$. Note that $(AB)^* = B^*A^*$. The matrix A is called *invertible* if there exists a matrix B such that $AB = I$. If it exists then this matrix is unique and will be called the *inverse* of A, denoted A^{-1}. Besides the usual product of matrices, we will also make use of the Hadamard product, written $A \circ B$, where $(A \circ B)[i, j] = A[i, j]B[i, j]$.

For an n-dimensional vectors u, v note that u^*v and uv^* are very different creatures—the first is a scalar and the second is an n-by-n matrix. The quantity u^*v is known as the *inner product* of u, v. Two vectors are called *orthogonal* if $u^*v = 0$. We will make frequent use of the ℓ_2 vector norm, defined as $\|v\| = \sqrt{v^*v}$. A set of vectors $v_1, \ldots, v_m$ will be called an *orthonormal set* if v_i, v_j are orthogonal for $i \neq j$ and each v_i has ℓ_2 norm one.

A set of vectors $v_1, \ldots v_m$ is called *linearly independent* if the only solution to $\sum_i \alpha_i v_i = (0, \ldots, 0)$ is to set $\alpha_i = 0$ for all $1 \leq i \leq m$. Written in matrix notation, this says that if A is the matrix whose i^{th} column is v_i, the only solution to $Au = 0$ is $u = (0, \ldots, 0)$. The *span* of a set of vectors $v_1, \ldots, v_m$ is the set $\{w :$

$w = \sum_i \alpha_i v_i, \alpha_i \in \mathbb{C}\}$. The Cauchy–Schwarz inequality says that $|u^* v| \leq \|u\|\|v\|$.

B.2 Matrix and Vector Norms

In this section we look at ways to measure the "size" of a matrix—norms. By ignoring the array-like structure of a $m \times n$ matrix A, we can also think of it as a vector of dimension mn. In this way, all notions which apply to vectors, in particular vector norms, can directly be applied to matrices.

B.2.1. Definition. A function $\|\cdot\| : \mathbb{C}^d \to \mathbb{R}$ is called a vector norm if for all $x, y \in V$ and scalars $c \in \mathbb{C}$:

1. $\|x\| \geq 0$
2. $\|x\| = 0$ if and only if $x = 0$
3. $\|cx\| = |c|\|x\|$
4. $\|x + y\| \leq \|x\| + \|y\|$

B.2.2. Example. Examples of vector norms include the ℓ_p norms for $1 \leq p < \infty$.

$$\|v\|_p = \left(\sum_i |v[i]|^p \right)^{1/p}.$$

Taking the limit as p goes to ∞ we have the ℓ_∞ norm:

$$\|v\|_\infty = \max_i |v[i]|.$$

Matrices possess a natural multiplication operation which is lost when they are considered simply as vectors. Thus we are led to the definition of a matrix norm which also takes into account matrix multiplication.

B.2.3. Definition. A function $\|\cdot\| : M_n \to \mathbb{R}$ is called a matrix norm if for all $A, B \in M_n$ and scalars c:

1. $\|A\| \geq 0$
2. $\|A\| = 0$ if and only if $A = 0$
3. $\|cA\| = |c|\|A\|$
4. $\|A + B\| \leq \|A\| + \|B\|$
5. $\|AB\| \leq \|A\|\|B\|$

The ℓ_2 vector norm of a matrix turns out to also be a matrix norm. In the context of matrices, this is referred to as the Frobenius norm, and is denoted $\|A\|_F$. To see that this is a norm it is clear that we only have to check is property 5. Letting A_i denote the i^{th} row of A, and similarly for B we find:

$$\begin{aligned}\|AB\|_F^2 &= \sum_{i,j} |A_i^* B_j|^2 \\ &\leq \sum_{i,j} \|A_i\|^2 \|B_j\|^2 = \|A\|_F^2 \|B\|_F^2.\end{aligned}$$

There is a general way to define a matrix norm from a vector norm. For a vector norm $\|\cdot\|$ we call the matrix function

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

the matrix norm *induced* by the vector norm $\|\cdot\|$. Again the only case which does not directly follow from the fact that $\|\cdot\|$ is a vector norm is property 5. To check this, notice that

$$\begin{aligned}\|A\|\|B\| &= \max_{y \neq 0} \frac{\|Ay\|}{\|y\|} \max_{x \neq 0} \frac{\|Bx\|}{\|x\|} \\ &\geq \max_{x \neq 0, Bx \neq 0} \frac{\|ABx\|}{\|Bx\|} \frac{\|Bx\|}{\|x\|} \\ &= \max_{x \neq 0} \frac{\|ABx\|}{\|x\|} = \|AB\|.\end{aligned}$$

An example of a vector induced matrix norm is the spectral norm, which is induced by the ℓ_2 vector norm. The spectral norm plays an important role in this thesis and we will study it in more detail in the sequel.

Two other matrix norms we will use in this thesis are the maximum absolute row sum norm and the maximum absolute column sum norm.

B.2.4. Definition. Let A be an arbitrary m-by-n matrix. Then

$$\begin{aligned}\|A\|_1 &= \max_j \sum_{i=1}^m |A[i,j]| \\ \|A\|_\infty &= \max_i \sum_{j=1}^n |A[i,j]|\end{aligned}$$

B.3 Rank

One of the most basic quantities associated with a matrix is its rank. The rank of a matrix A, written $\mathrm{rk}(A)$, is the size of a largest set of linearly independent columns of A. Whether we take columns or rows here is not important as the number of linearly independent rows and columns is the same. Similarly, it can be shown that $\mathrm{rk}(A) = \mathrm{rk}(A^*)$.

Suppose that A is an m-by-n matrix with $m \geq n$ and rank n. As the columns of A are linear independent, the only solution to the equation $Av = 0$ is $v = \vec{0}$. This implies that A is injective as if $Au = Av$ for some $u \neq v$ then $A(u - v) = 0$ and $u - v \neq \vec{0}$, a contradiction. If $m = n$ then A is also surjective and thus invertible.

This reasoning can be extended to show that if A is an m-by-n matrix with $m \geq n$ and rank k, then the set $\{v : Av = 0\}$ is spanned by a set of $n - k$ vectors.

B.4 Unitary Matrices and Orthonormal Sets

An orthonormal set of vectors is nice to work with as it has such a clear structure. The Gram–Schmidt process gives a way to construct from any set of linearly independent vectors a set of orthonormal vectors with the same span.

B.4.1. Fact (Gram–Schmidt orthonormalization). *Let $S = \{u_1, \ldots, u_m\}$ be a set of linearly independent vectors. There exists an orthonormal set $v_1, \ldots, v_m$ with the same span as S.*

Proof: We prove by induction. Let $v_1 = u_1/\|u_1\|$. Then it is clear that v_1 has unit length and has the same span as u_1.

Now suppose that the orthonormal set $v_1, \ldots, v_{k-1}$ has been constructed and has the same span as $u_1, \ldots, u_{k-1}$. Let

$$v'_k = u_k - \sum_{i=1}^{k-1} (u_k^* v_i) v_i.$$

In this way, v'_k is orthogonal to each v_i with $i < k$. We now set $v_k = v'_k/\|v'_k\|$. Thus $v_1, \ldots, v_k$ form an orthonormal with the same span as $u_1, \ldots, u_k$, as v_k was formed from a linear combination of $u_1, \ldots, u_k$. □

An important class of matrices are those whose columns form an orthonormal set. These are known as unitary matrices.

B.4.2. Definition. A matrix U is called unitary if $U^*U = I$

A nice property of unitary matrices is that they preserve inner product.

B.4.3. Fact. Let U be unitary. Then $(Uv)^*Uw = v^*w$ for all v, w.

Proof: Notice that $(Uv)^*Uw = v^*U^*Uw = v^*w$. □

B.5 Eigenvalues

For an n-by-n matrix A, if $\lambda \in \mathbb{C}$ and $0 \neq v \in \mathbb{C}^n$ satisfy $Av = \lambda v$ then λ is called an eigenvalue of A, and similarly v is called an eigenvector. We will write the eigenvalues of A in nonincreasing order, $\lambda_1 \geq \ldots \geq \lambda_n$. We will also make use of a functional notation where $\lambda_i(A)$ denotes the i^{th} largest eigenvalue of A.

When do two matrices A and B have the same eigenvalues? A sufficient, but not necessary, condition for this to hold is for A and B to be *similar*. Two matrices A, B are called similar if there exists an invertible matrix S such that $A = S^{-1}BS$. Notice that if A and B are similar and $Av = \lambda v$ then $BS = \lambda Sv$, and so they have the same eigenvalues.

The concept of similarity is important for the following fact:

B.5.1. Fact. Let A be a m-by$n$ matrix and B be a n-by-m matrix with $m \leq n$. Then BA has the same eigenvalues as AB with an additional $n - m$ eigenvalues equal to zero.

Proof: The proof is easy if $m = n$ and at least one of A, B is invertible. For if A is invertible, then $A^{-1}ABA = BA$ and thus A and B are similar and have the same eigenvalues. We now reduce the general case to this simple case by enlarging to $m + n$ square matrices. Notice the following identities

$$\begin{aligned} \begin{pmatrix} AB & 0 \\ B & 0 \end{pmatrix} \begin{pmatrix} I & A \\ 0 & I \end{pmatrix} &= \begin{pmatrix} AB & ABA \\ B & BA \end{pmatrix} \\ \begin{pmatrix} I & A \\ 0 & I \end{pmatrix} \begin{pmatrix} 0 & 0 \\ B & BA \end{pmatrix} &= \begin{pmatrix} AB & ABA \\ B & BA \end{pmatrix} \end{aligned}$$

Now as the $m + n$-by-$m + n$ matrix

$$\begin{pmatrix} I & A \\ 0 & I \end{pmatrix}$$

is nonsingular—it has all eigenvalues equal to $+1$—these identities imply that the two square $m + n$ matrices

$$C_1 = \begin{pmatrix} AB & 0 \\ B & 0 \end{pmatrix} \text{ and } C_2 = \begin{pmatrix} 0 & 0 \\ B & BA \end{pmatrix}$$

are similar and thus have the same eigenvalues. Notice that the eigenvalues of C_1 are those of AB together with n zeros and the eigenvalues of C_2 are those of BA together with m zeros. The assertion follows. □

A diagonal matrix D has a particularly simple form, satisfying $D[i, j] = 0$ if $i \neq j$. Notice that the eigenvalues of a diagonal matrix are simply its diagonal entries. We call a matrix *diagonalizable* if it is similar to a diagonal matrix. Notice that a n-by-n matrix A is diagonalizable if and only if it has a set of n linearly independent eigenvectors. Letting S be the matrix with these eigenvectors as its columns we see that $AS = DS$ where D is a diagonal matrix with the eigenvalues of A on its diagonal. The invertibility of S is assured as its columns are linear independent, thus we have $A = S^{-1}DS$. It is particularly nice if a matrix is diagonalizable by a unitary matrix, as then it has an orthonormal set of eigenvectors. We note the class of matrices which have this property:

B.5.2. Fact. The following are equivalent:

1. A is diagonalizable by a unitary matrix
2. $A^*A = AA^*$. We call such a matrix normal.

B.6 Singular Values

In this thesis we will quite frequently deal with nonsquare matrices. As this might not be as familiar to the reader as the square case, we review some things one should be careful of in dealing with nonsquare matrices. The first of these is that the concept of eigenvalue no longer makes sense when working with nonsquare matrices. A natural way to extend the concept of eigenvalues to nonsquare matrices is through what are called singular values.

B.6.1. Definition. For a matrix A, not necessarily square, we define the i^{th} singular value of A to be:

$$\sigma_i(A) = \sqrt{\lambda_i(A^*A)}.$$

B.6.2. Remark. By Fact B.5.1 A^*A and AA^* have the same nonzero eigenvalues, thus is not crucial in this definition that we take A^*A instead of AA^*. Fact B.5.1 also has the consequence that for a m-by-n matrix A with $m \leq n$ that $\sigma_i(A) = \sigma_i(A^*)$ for all $1 \leq i \leq m$.

Notice that if A is Hermitian then A^*A is simply A^2 and so the singular values and eigenvalues of A agree, up to sign. This is not the case in general as can be seen by the matrix $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ which has eigenvalues 1 and 0 and singular values $\sqrt{2}$ and 0.

We now look at some alternative characterizations of the spectral norm. The first of these says that the spectral norm of A is the largest singular value of A.

B.6.3. Fact. The following are equivalent:

1. $\|A\| = \max\{\sqrt{\lambda} : \lambda \text{ is an eigenvalue of } A^*A\}$

2. $\|A\| = \max_{x \neq 0} \|Ax\|/\|x\|$

3. $\|A\| = \max_{x,y \neq 0} |y^*Ax|/\|x\|\|y\|$

Proof: (1) $\leq$ (2): Let v_1 be an eigenvector of unit length corresponding to the largest eigenvalue λ_1 of A^*A. Then

$$\|Av_1\|^2 = v^*A^*Av = \lambda_1.$$

(1) $\geq$ (2): We show that $\lambda_1(A^*A) \geq \max_x \|Ax\|^2$. As A^*A is normal, it has an orthonormal set of eigenvectors $v_1, \ldots, v_n$ corresponding to eigenvalues $\lambda_1 \geq \ldots \lambda_n$. Let v be an arbitrary vector with unit ℓ_2 norm. We can express v as $v = \sum_i \alpha_i v_i$ where $\sum_i |\alpha_i|^2 = 1$. Now

$$\|Av\|^2 = v^*A^*Av = \sum_i \lambda_i |\alpha_i|^2 \leq \lambda_1.$$

(2) $\leq$ (3): Let x maximize $\|Ax\|$. Set $y = Ax$. Then we have

$$y^*Ax = x^*A^*Ax/\|Ax\| = \|Ax\|$$

(3) $\leq$ (2): By the Cauchy–Schwarz inequality, $|y^*Ax| \leq \|y\|\|Ax\|$. □

Finally, we will make use of the following upper bound on the spectral norm.

B.6.4. Fact. Let A be an arbitrary m-by-n matrix.

$$\|A\|^2 \leq \|A\|_1 \|A\|_\infty.$$

Bibliography

[ABK$^+$02] E. Allender, H. Buhrman, M. Koucky, D. van Melkebeek, and D. Ronneburger. Power from random strings. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 669–678. IEEE, 2002. 69

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 75–83. IEEE, 1978. 14

[AF05] L. Antunes and L. Fortnow. Personal communication, 2005. 25, 62

[Amb02] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64:750–767, 2002. 114, 119, 133, 134

[Amb03] A. Ambainis. Polynomial degree vs. quantum query complexity. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 230–239. IEEE, 2003. 114, 120, 122, 136, 138

[And87] A. E. Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of Π-schemes. *Moscow Univ. Math. Bull.*, 42(1):63–65, 1987. 103

[AS04] S. Aaronson and Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM*, 51(4):595 – 605, 2004. 137

[AUY83] A. Aho, J. Ullman, and M. Yannakakis. On notions of information transfer in VLSI circuits. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 133–139. ACM, 1983. 107

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on the Theory of Computing*, pages 421–429. ACM, 1985. 30, 67, 72, 73

[Bas00] T. A. Bass. *The Eudaemonic Pie.* Houghton Mifflin, 2000. 5

[BBBV97] C.H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26:1510–1523, 1997. 119

[BBF98] R. Beigel, H. Buhrman, and L. Fortnow. NP might not be as easy as detecting unique solutions. In *Proceedings of the 30th ACM Symposium on the Theory of Computing*, pages 203–208. ACM, 1998. 80

[BF95] H. Buhrman and L. Fortnow. Distinguishing complexity and symmetry of information. Technical Report TR-95-11, Department of Computer Science, The University of Chicago, 1995. 70, 74

[BFL02] H. Buhrman, L. Fortnow, and S. Laplante. Resource bounded Kolmogorov complexity revisited. *SIAM Journal on Computing*, 31(3):887–905, 2002. 32, 37

[BG81] C. Bennett and J. Gill. Relative to a random oracle, $P^A \neq NP^A \neq co-NP^A$ with probability one. *SIAM Journal on Computing*, 10:96–113, 1981. 30

[BHT97] G. Brassard, P. Høyer, and A. Tapp. Quantum algorithm for the collision problem. *ACM SIGACT News (Cryptology column)*, 28:14–19, 1997. 137

[BLM00] H. Buhrman, S. Laplante, and P. Bro Miltersen. New bounds for the language compression problem. In *Proceedings of the 15th IEEE Conference on Computational Complexity*, pages 126–130. IEEE, 2000. 37, 57, 59

[BLvM05] H. Buhrman, T. Lee, and D. van Melkebeek. Language compression and pseudorandom generators. *Computational Complexity*, 14:247–274, 2005. 37, 43

[Bop89] R. Boppana. Amplification of probabilistic boolean formulas. *Advances in Computing Research*, 5(4):27–45, 1989. 101

[BSS03] H. Barnum, M. Saks, and M. Szegedy. Quantum decision trees and semidefinite programming. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, pages 179–193, 2003. 114, 120, 121

[BW02] H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288:21–43, 2002. 102

[CHLL97] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering Codes.* North-Holland, Amsterdam, 1997. 82, 84

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* MIT Press, 1990. 85

[DGHS00] E. Dantsin, A. Goerdt, E. Hirsch, and U. Schöning. Deterministic algorithms for k-SAT based on covering codes and local search. In *Proceedings of the 27th International Colloquium On Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 236–247. Springer-Verlag, 2000. 86

[DHM04] P. Diaconis, S. Holmes, and R. Montgomery. Dynamical bias in the coin toss. http://www-stat.stanford.edu/˜cgates/PERSI, 2004. 3

[DR82] A. G. Dyachkov and V. V. Rykov. Bounds on the length of disjunctive codes. *Problemy Peredachi Informatsii*, 18:7–13, 1982. In Russian. 57

[DZ97] M. Dubiner and U. Zwick. Amplification by read-once formulas. *SIAM Journal on Computing*, 26(1):15–38, 1997. 101

[Fan49] K. Fan. On a theorem of Weyl concerning eigenvalues of linear transformations I. *Proceedings of the national academy of sciences of the USA*, 35:652–655, 1949. 114

[FF93] J. Feigenbaum and L. Fortnow. On the random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22:994–1005, 1993. 48

[FGM$^+$89] M. Fürer, O. Goldreich, Y. Mansour, M. Sipser, and S. Zachos. On completeness and soundness in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 429–442. JAI Press, Greenwich, 1989. 30, 52

[FK96] L. Fortnow and M. Kummer. On resource-bounded instance complexity. *Theoretical Computer Science A*, 161:123–140, 1996. 79

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984. 44

[Gol01] O. Goldreich. Lecture notes on pseudorandomness–Part I, 2001. available online at http://www.wisdom.weizmann.ac.il/˜oded/c-indist.html. 5

[GS89] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 73–90. JAI Press, Greenwich, 1989. 64

[GS91] A. Goldberg and M. Sipser. Compression and ranking. *SIAM Journal on Computing*, 20:524–536, 1991. 54

[GV99] O. Goldreich and S. Vadhan. Comparing entropies in statistical zero-knowlede with applications to the structure of SZK. In *Proceedings of the 14th IEEE Conference on Computational Complexity*, pages 54–73. IEEE, 1999. 31, 65

[Hås98] J. Håstad. The shrinkage exponent is 2. *SIAM Journal on Computing*, 27:48–64, 1998. 101, 103

[HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 64

[HJ99] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1999. 118

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965. 27

[HS66] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13:533–546, 1966. 22

[IM02] K. Iwama and H. Morizumi. An explicit lower bound of $5n - o(n)$ for boolean circuits. In *Proceedings of the 27th International Symposium on Mathematical Foundations of Computer Science*, pages 353–364, 2002. 100

[ISW00] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-random generators with optimal seed length. In *Proceedings of the 32nd ACM Symposium on the Theory of Computing*, pages 1–10. ACM, 2000. 83

[JKS03] T. Jayram, R. Kumar, and D. Sivakumar. Two applications of information complexity. In *Proceedings of the 35th ACM Symposium on the Theory of Computing*, pages 673–682. ACM, 2003. 134

[JSV97] T. Jiang, J. Seiferas, and P. Vitányi. Two heads are better than two tapes. *Journal of the ACM*, 44(2):237–256, 1997. 19, 69

[Kel86] J.B. Keller. The probability of heads. *American Mathematical Monthly*, 93:191–197, 1986. 3

[Khr71] V.M. Khrapchenko. Complexity of the realization of a linear function in the case of Π-circuits. *Math. Notes Acad. Sciences*, 9:21–23, 1971. 104

[KI03] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th ACM Symposium on the Theory of Computing*, pages 355–364. ACM, 2003. 1

[KKN95] M. Karchmer, E. Kushilevitz, and N. Nisan. Fractional covers and communication complexity. *SIAM Journal on Discrete Mathematics*, 8(1):76–92, 1995. 107, 109, 110, 113

[Kla04] H. Klauck. One-way communication complexity and the Nečiporuk lower bound on formula size. Technical Report 0111062, cs.CC arXiv, 2004. 102, 110

[KLO96] E. Kushilevitz, N. Linial, and R. Ostrovsky. The linear array conjecture of communication complexity is false. In *Proceedings of the 28th ACM Symposium on the Theory of Computing*. ACM, 1996. 139

[Ko82] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Information Processing Letters*, 14(1):39–43, 1982. 65

[Kou93] E. Koutsoupias. Improvements on Khrapchenko's theorem. *Theoretical Computer Science*, 116(2):399–403, 1993. 105

[KS99] R. Kumar and D. Sivakumar. Proofs, codes, and polynomial-time reducibilities. In *Proceedings of the 14th IEEE Conference on Computational Complexity*, pages 46–53. IEEE, 1999. 42

[KSV03] M. Krivelevich, B. Sudakov, and V. Vu. Covering codes with improved density. *IEEE Transactions on Information Theory*, 49:1812–1815, 2003. 84

[Kur87] S. Kurtz. A note on randomized polynomial time. *sicomp*, 16(5), 1987. 30

[KvM02] A. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002. 39

[KW88] M. Karchmer and A. Wigderson. Monotone connectivity circuits require super-logarithmic depth. In *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pages 539–550, 1988. 105

[Lev73] L. A. Levin. Universal search problems. *Problems Information Transmission*, 9(3):265–266, 1973. 71

[LM93] L. Longpré and S. Mocas. Symmetry of information and one-way functions. *Information Processing Letters*, 46(2):95–100, 1993. 69, 70

[LM04] S. Laplante and F. Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 294–304. IEEE, 2004. 114, 120, 121, 133

[Lov93] L. Lovasz. *Combinatorial Problems and Exercises.* North-Holland, 1993. 3

[LR01] O. Lachish and R. Raz. Explicit lower bound of 4.5n - o(n) for Boolean circuits. In *Proceedings of the 33rd ACM Symposium on the Theory of Computing*, pages 399–408. ACM, 2001. 100

[LV92a] M. Li and P.M.B. Vitányi. Inductive reasoning and Kolmogorov complexity. *Journal of Computer and System Sciences*, 44(2):343–384, 1992. 63

[LV92b] M. Li and P.M.B. Vitányi. Worst case complexity is equal to average case complexity under the universal distribution. *Information Processing Letters*, 42:145–149, 1992. 63

[LV97] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, New York, second edition, 1997. 4, 8, 69, 83

[LW95] L. Longpré and O. Watanabe. On symmetry of information and polynomial time invertibility. *Information and Compuation*, 121(1):14–22, 1995. 70, 73

[MO03] E. Mossell and R. O'Donnell. On the noise sensitivity of monotone functions. *Random Structures and Algorithms*, 23(3):333–350, 2003. 134

[Nat95] B. Natarajan. Filtering random noise from deterministic signals via data compression. *IEEE transactions on signal processing*, 43(11):2595–2605, 1995. 81

[Neč66] E. I. Nečiporuk. A Boolean function. *Soviet Mathematics–Doklady*, 7:999–1000, 1966. 110

[Nis91] N. Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. 134

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994. 1, 30, 39, 41, 43

[O'D02] R. O'Donnell. Hardness amplification within NP. In *Proceedings of the 34th ACM Symposium on the Theory of Computing*, pages 751–760. ACM, 2002. 134

[OW92] M. Overton and R. Womersley. On the sum of the largest eigenvalues of a symmetric matrix. *SIAM Journal on Matrix Analysis Applications*, 13(1):41–45, 1992. 114

[Ram29] F. Ramsey. On a problem of formal logic. *Proceedings of the London Mathematical Society*, 30(2):264–286, 1929. 13

[Raz90] A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10(1):81–93, 1990. 124

[Ron04] D. Ronneburger. Personal communication, 2004. 71

[RRV02] R. Raz, O. Reingold, and S. Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *Journal of Computer and System Sciences*, 65(1):97–128, 2002. 40, 41, 42, 43

[Sch83] U. Schürfeld. New lower bounds on the formula size of Boolean functions. *Acta Informatica*, 19(2):183–194, 1983. 133

[Sha02] R. Shaltiel. Recent developments in explicit construction of extractors. *Bulletin of the European Association for Theoretical Computer Science*, 77:67–95, 2002. 39

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 330–335. ACM, 1983. 20, 28

[Spi71] P. Spira. On time-hardware complexity tradeoffs for Boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences*, pages 525–527. Western Periodicals Company, North Hollywood, 1971. 29, 101

[ŠS05] R. Špalek and M. Szegedy. All quantum adversary methods are equivalent. In *Proceedings of the 32th International Colloquium On Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 1299–1311. Springer-Verlag, 2005. quant-ph/0409116. 114, 120, 128, 133

[STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001. 42

[SU01] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 648–657. IEEE, 2001. 41, 48

[Sub61] B. A. Subbotavskaya. Realizations of linear functions using $\{+, *, -\}$. *Soviet Mathematics–Doklady*, 2:110–112, 1961. 103

[Sud97] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997. 42

[Sze03] M. Szegedy. An $O(n^{1.3})$ quantum algorithm for the triangle finding problem. Technical report, 2003. quant-ph/0310134. 133

[Tre01] L. Trevisan. Construction of extractors using pseudo-random generators. *Journal of the ACM*, 48(4):860–879, 2001. 39, 42, 43

[TSUZ01] A. Ta-Shma, C. Umans, and D. Zuckerman. Loss-less condensers, unbalanced expanders, and extractors. In *Proceedings of the 33rd ACM Symposium on the Theory of Computing*, pages 143–152. ACM, 2001. 46

[TSZS01] A. Ta-Shma, D. Zuckerman, and S. Safra. Extractors from Reed-Muller codes. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 638–647. IEEE, 2001. 41

[TVZ04] L. Trevisan, S. Vadhan, and D. Zuckerman. Compression of samplable sources. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 1–15. IEEE, 2004. 54, 61

[Uma03] C. Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67:419–440, 2003. 41

[Val84] L.G. Valiant. Short monotone formulae for the majority function. *Journal of Algorithms*, 5:363–366, 1984. 101

[vM19] R. von Mises. Grundlagen der Wahrscheinlichkeitsrechnung. *Mathemat. Zeitsch.*, 5:52–99, 1919. 3

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986. 29, 36, 79, 80

[VV04a] N. Vereschagin and P. Vitányi. Algorithmic rate-distortion theory. http://arxiv.org/abs/cs.IT/0411014, 2004. 86

[VV04b] N.K. Vereshchagin and P.M.B. Vitányi. Kolmogorov's structure functions and model selection. *IEEE Trans. Inform. Theory*, 50(12):3265–3290, 2004. 69

[Wee04] H. Wee. On pseudoentropy versus compressibility. In *Proceedings of the 19th IEEE Conference on Computational Complexity.* IEEE, 2004. 80

[Zha05] S. Zhang. On the power of Ambainis's lower bounds. *Theoretical Computer Science*, 339(2–3):241–256, 2005. 114, 120, 133

[ZL70] A. Zvonkin and L. Levin. The complexity of finite objects and the algorithmic concepts of information and randomness. *Russian Mathematical Surveys*, 25:83–124, 1970. 69, 71

Samenvatting

Deel I: Kolmogorov Complexiteit

Toeval is iets waar iedereen kennis van heeft in het dagelijkse leven. De verandering van het weer, de worp van een muntje, de shuffle functie op een muziek speler—"toeval" is vaak te gebruiken in verband met deze dingen. Ondanks deze intuitieve kennis, of misschien juist daarom, hebben wiskundigen pas in de tweede helft van de twintigste eeuw een precieze definitie voor toeval gegeven.

In het begin van de jaren zestig, hebben drie onderzoekers Solomonoff, Kolmogorov en Chaitin onafhankelijk van elkaar een elegante wiskundig manier ontwikkeld om te zeggen wanneer een serie van gebeurtenissen (gezien als een rijtje nullen en enen) willekeurig is. Ze zeiden dat de complexiteit van een woord de grootte is van een kortste beschrijving voor het woord. Men kan hier denken aan een beschrijving in de vorm van een computerprogramma. Dit noemen we de Kolmogorov complexiteit van een woord en schrijven we $C(x \mid y)$ voor de grootte van een kortste programma voor x dat het woord y als invoer kan gebruiken. Men kan zien dat terwijl het woord

$$01010101010101010101010101010101010101010101010101$$

een lengte van 50 bits heeft, er een hele korte beschrijving voor dit woord is, namelijk "schrijf 25 keer 01". Aan de andere kant, een geheel willekeurig woord heeft geen structuur die gebruikt kan worden om het een korte beschrijving te geven. Dus noemen we een woord willekeurig als de kortste beschrijving van het woord minstens zo lang is als het woord zelf.

Meer dan enkel een definitie voor willekeurigheid, is Kolmogorov complexiteit geworden tot een algemeen hulpmiddel dat veel wordt gebruikt in de informatica. Bijna alle applicaties van Kolmogorov complexiteit maken gebruik van een van de volgende vier stellingen die we de 'vier pilaren' van Kolmogorov complexiteit noemen:

- Incompressibility: Voor elke n bestaat er een willekeurig woord van lengte n.

- Verzameling compressie: Elk lid van een berekenbare verzameling A heeft een beschrijving van lengte $\log |A|$
- Bron compressie: Elk woord x met positieve kans onder een berekenbare distributie P heeft een beschrijving van lengte $-\log P(x)$
- Symmetrie van informatie: De hoeveelheid infomatie in het woord x over het woord y is hetzelfde als de hoeveelheid informatie in het woord y over x. Met andere woorden, $C(x) - C(x \mid y) = C(y) - C(y \mid x)$.

Een nadeel van de Kolmogorov complexiteitstheorie is dat er geen algoritme bestaat die, gegeven een woord x, de complexiteit van x kan berekenen. Om dit probleem te omzeilen, ontwikkelde men *resource begrensde* Kolmogorov complexiteitstheorie. Bijvoorbeld, de polynomiale tijd Kolmogorov complexiteit van x is de lengte van een kortste programma dat x afdrukt in tijd polynomiaal in de lengte van x. Het doel van het eerste deel van dit proefschrift is het beantwoorden van de vraag: wat gebeurt er met de vier bovengenoemde pilaren in de resource begrensde variant?

De eerste pilaar werkt onveranderd in de resource begrensde variant, want het is moeilijker een woord te beschrijven in beperkte tijd dan zonder een tijd grens.

De resource begrensde variant wordt veel interssanter met de tweede pilaar. Een natuurlijke formulering van de tweede pilaar is: elk lid x van een verzameling A die in polynomiale tijd kan worden herkend, heeft een beschrijving van $\log |A|$ die x afdrukt in tijd polynomiaal in de lengte van x. Dit vermoeden is waarschijnlijk niet waar aangezien het de ineenstorting van de polynomiale hierarchie tot gevolg heeft, wat de meeste onderzoekers onvoorstelbaar achten. Maar een stap hoger in de polynomiaal hierarchie geldt het vermoeden wel: We laten zien dat elk lid x van een verzameling A die in niet-deterministisch polynomiale tijd kan worden herkend, een beschrijving heeft van lengte $\log |A|$ die x afdrukt in niet-deterministisch polynomiale tijd.

Een natuurlijk analogon van de derde pilaar in de resource begrensde variant zegt: elk woord x met positieve kans onder een polynomiale tijd berekenbare distributie P heeft een beschrijving van lengte $-\log P(x)$ die x afdrukt in polynomiale tijd. We bewijzen dat dit vermoeden als gevolg heeft dat gerandomiseerde algoritmen (die hun keuzes kunnen laten afhangen van muntworpen) efficient gesimuleerd kunnen worden door deterministische algoritmes (die geen muntjes kunnen werpen). Met andere woorden, BPP $\neq$ EXP. Onlangs lieten Antunes and Fortnow zien dat de omgekeerde implicatie ongeveer geldt. Hiermee hebben we een bijna compleet beeld van de derde pilaar in de resource begrensde variant.

Ten slotte laten we zien hoe het principe van symmetrie van informatie uitpakt in de resource begrensde variant. Dit principe wordt heel moeilijk want het standaard bewijs maakt gebruik van zowel verzameling compressie als bron compressie. We laten een zwakkere vorm van symmetrie van informatie zien die zegt: de polynomiale tijd complexiteit van het paar (x, y) is groter dan de gerandomiseerde niet-deterministische complexiteit van x plus de gerandomiseerde niet-deterministische complexiteit van y met x als invoer. Dit resultaat kan niet verbetered worden zonder zogeheten niet-relativerende technieken. We laten een orakel zien waarvan de niet-deterministische complexiteit van het paar (x, y) bijna twee keer groter is dan de niet-deterministische complexiteit van x plus de niet-deterministische complexiteit van y met x als invoer.

Deel II: Formulegrootte

Een van de meest beroemde, belangrijke en moeilijke problemen binnen de computationele complexiteitstheorie betreft de vraag of P even krachtig is als NP. P is de klasse van problemen die opgelost kunnen worden in tijd polynomiaal in de lengte van de probleembeschrijving; NP is de klasse van problemen waarvan in polynomiale tijd gecontroleerd kan worden of een oplossing correct is. Tegenwoordig geloven de meeste onderzoekers dat P en NP niet gelijk zijn. Om dit te bewijzen, moet men laten zien dat er een probleem bestaat in NP dat niet in polynomiale tijd kan worden opgelost. Dit werd vaak onderzocht door het probleem te representeren als een circuit dat bestaat uit AND, OR en NOT poorten. De grootte van het circuit geeft vervolgens een ondergrens voor de benodigde rekentijd. Het beste huidige resultaat laat zien dat er een probleem is in NP dat niet kan worden opgelost met een circuit van grootte kleiner dan 5n, waar n de grootte van de probleembeschrijving is. Een hogere ondergrens op de grootte van het circuit blijkt erg moeilijk te bewijzen. Om die reden beperken wij ons tot circuits waarvan de poorten slechts één uitgang hebben. Een dergelijk circuit wordt een formule genoemd. De hoogst bekende ondergrens op de grootte van een formule voor een functie in NP is n^3.

We geven een nieuwe algebraische methode om de formulegrootte van onderen te begrenzen. Met deze methode kunnen we de n^3-ondergrens niet verbeteren, maar wij generaliseren verschillende methoden die in de literatuur worden beschreven en we demonstreren hoe die methoden met elkaar in verband staan. We geven ook een voorbeeld waar onze methoden een sterkere ondergrens kunnen geven dan andere methoden. Het meest interessante gevolg van ons werk is misschien het verrassende verband tussen de formulegrootte van een functie en de complexiteit van die functie in een heel ander model, namelijk de quantum query-complexiteit. Onze resultaten geven aanleiding tot het vermoeden dat het kwadraat van de quantum query complexiteit van een functie een ondergrens is voor de formulegrootte.

Abstract

Part I: Kolmogorov Complexity

Randomness is a concept familiar to all of us in our daily lives. From changing weather, to the flip of a coin, to the shuffle function on a music player—random is frequently used to describe the behavior of things around us. Despite this familiarity, or perhaps because of it, randomness eluded a precise mathematical definition until the second half of the twentieth century.

In the early 1960's, independently and within a few years of each other, Solomonoff, Kolmogorov, and Chaitin all developed an elegant way to capture when a sequence of events (viewed as a string with letters 0 and 1) is random. They proposed the following definition: the complexity of a string is the length of a shortest description of that string. Here a description can be thought of as a computer program. We will refer to this measure as the Kolmogorov complexity of the string, and write $C(x \mid y)$ for the length of a shortest program for x which is allowed to make use of the advice string y. While the string 01010101010101010101010101010101010101010101010101 has 50 symbols, it has a much shorter description saying "write 25 times the string 01". On the other hand, a random string will have no structure which would allow a description of length shorter than the length of the string itself. Thus we call a string random if its Kolmogorov complexity is at least as large as its length.

Far beyond its initial purpose, Kolmogorov complexity has become an important tool in theoretical computer science, witnessing many diverse applications. Almost all of the applications use at least one of the following four fundamental theorems, which we call the 'four pillars' of Kolmogorov complexity:

- Incompressibility: There is a random string of length n, for each n.

- Language compression: Any element from a computable set A can be given a description of size about $\log |A|$.

- Source compression: Any element x in the support of a computable probability distribution P can be given a description of size about $-\log P(x)$.
- Symmetry of information: The information which a string x contains about a string y is about the same as that which y contains about x. In symbols: $C(x) - C(x \,|\, y) = C(y) - C(y \,|\, x)$.

A drawback to Kolmogorov complexity is that it is uncomputable, and this sometimes limits its range of applicability in computational complexity. One way to scale down the theory into the feasible domain is to require that the program which prints the string x does so in time which is polynomial in the length of x. The main goal of the first part of this thesis is to see what analogues, if any, of the four pillars hold in this resource bounded setting.

The first pillar holds unchanged in the resource bounded setting, as adding restrictions to a program can only make it harder to describe a string.

Things get much more interesting beginning with the second pillar. The most natural resource bounded analog of the second pillar is the statement: every element x in a set A decidable in polynomial time has a description of length $\log |A|$ from which x can be generated in polynomial time. This statement is unlikely to hold, however, as it implies the polynomial hierarchy collapses. We show that going up one step higher in the polynomial hierarchy, however, we can find an analog of the second pillar. That is, we are able to show every element x in a set A which can be decided in nondeterministic polynomial time can be given a description of length about $\log |A|$ from which x can be generated in nondeterministic polynomial time.

The most natural analogue of the third pillar would be: any element in the support of a polynomial time computable probability distribution P can be given a description of length about $-\log P(x)$. We show that such a statement implies that randomized algorithms (those which can make decisions based on the outcome of a coin flip) can be simulated by deterministic algorithms (which cannot flip coins). Precisely, we show this implies BPP $\neq$ EXP. Recently, Antunes and Fortnow were able to prove a weak converse to this statement: under a derandomization assumption the polynomial time version of the third pillar holds. With these results we have a fairly complete picture of the third pillar in the resource bounded setting.

Finally, we turn to symmetry of information which is perhaps the most tricky of the four pillars in the resource bounded domain. To prove symmetry of information seems to require the ability to do both language compression and source compression. We are only able to prove a weaker version of symmetry of information which says that the polynomial time complexity of the pair (x, y) is larger than the randomized nondeterministic complexity of x plus the randomized nondeterministic complexity of y given x. This result is tight with respect to relativizing techniques—we give an oracle where even the nondeterministic complexity of (x, y) is nearly twice as large as the nondeterministic complexity of x plus the nondeterministic complexity of y given x.

Part II: Formula Size Lower Bounds

One of the most famous, important, and yes, most difficult problems in complexity theory is the question of whether or not P is equal to NP. P is the set of all problems that can be solved in time polynomial in the length of the problem description; NP is the set of problems which can be efficiently verified, meaning that if someone gives you the solution to the problem you can indeed check that it is a proper solution in polynomial time. Currently, it is widely believed that P is not equal to NP. To prove this one must show a lower bound, that is give an example of a problem in NP that requires more than polynomial time to solve. A well-studied approach to doing this is through circuit complexity, where a circuit consists of AND, OR, and NOT gates. The current best lower bound on the size of a circuit required to compute a function in NP is 5n. As the task of proving larger circuit lower bounds seems extremely difficult, we look instead at the weaker model of formula size, where a formula is simply a circuit where every gate has exactly one outgoing wire. The current best lower bound for formula size is n^3.

We give a new algebraic approach to proving formula size lower bounds. We are not able to improve on the best n^3 lower bound; we are, however, able to simultaneously generalize several techniques from the literature and show them as part of an overarching theory. We also give some concrete examples of functions for which our new method is able to provably do better than previous methods. Perhaps the most interesting consequence of our results, however, is that it gives evidence for a connection between the formula size of a function and its complexity in a very different model, namely its quantum query complexity. In fact, our results can be taken as evidence for the provocative conjecture that the square of the quantum query complexity of a function lower bounds its formula size.

Titles in the ILLC Dissertation Series:

ILLC DS-2001-01: **Maria Aloni**
Quantification under Conceptual Covers

ILLC DS-2001-02: **Alexander van den Bosch**
Rationality in Discovery - a study of Logic, Cognition, Computation and Neuropharmacology

ILLC DS-2001-03: **Erik de Haas**
Logics For OO Information Systems: a Semantic Study of Object Orientation from a Categorial Substructural Perspective

ILLC DS-2001-04: **Rosalie Iemhoff**
Provability Logic and Admissible Rules

ILLC DS-2001-05: **Eva Hoogland**
Definability and Interpolation: Model-theoretic investigations

ILLC DS-2001-06: **Ronald de Wolf**
Quantum Computing and Communication Complexity

ILLC DS-2001-07: **Katsumi Sasaki**
Logics and Provability

ILLC DS-2001-08: **Allard Tamminga**
Belief Dynamics. (Epistemo)logical Investigations

ILLC DS-2001-09: **Gwen Kerdiles**
Saying It with Pictures: a Logical Landscape of Conceptual Graphs

ILLC DS-2001-10: **Marc Pauly**
Logic for Social Software

ILLC DS-2002-01: **Nikos Massios**
Decision-Theoretic Robotic Surveillance

ILLC DS-2002-02: **Marco Aiello**
Spatial Reasoning: Theory and Practice

ILLC DS-2002-03: **Yuri Engelhardt**
The Language of Graphics

ILLC DS-2002-04: **Willem Klaas van Dam**
On Quantum Computation Theory

ILLC DS-2002-05: **Rosella Gennari**
Mapping Inferences: Constraint Propagation and Diamond Satisfaction

ILLC DS-2002-06: **Ivar Vermeulen**
A Logical Approach to Competition in Industries

ILLC DS-2003-01: **Barteld Kooi**
Knowledge, chance, and change

ILLC DS-2003-02: **Elisabeth Catherine Brouwer**
Imagining Metaphors: Cognitive Representation in Interpretation and Understanding

ILLC DS-2003-03: **Juan Heguiabehere**
Building Logic Toolboxes

ILLC DS-2003-04: **Christof Monz**
From Document Retrieval to Question Answering

ILLC DS-2004-01: **Hein Philipp Röhrig**
Quantum Query Complexity and Distributed Computing

ILLC DS-2004-02: **Sebastian Brand**
Rule-based Constraint Propagation: Theory and Applications

ILLC DS-2004-03: **Boudewijn de Bruin**
Explaining Games. On the Logic of Game Theoretic Explanations

ILLC DS-2005-01: **Balder David ten Cate**
Model theory for extended modal languages

ILLC DS-2005-02: **Willem-Jan van Hoeve**
Operations Research Techniques in Constraint Programming

ILLC DS-2005-03: **Rosja Mastop**
What can you do? Imperative mood in Semantic Theory

ILLC DS-2005-04: **Anna Pilatova**
A User's Guide to Proper names: Their Pragmatics and Semanics

ILLC DS-2006-01: **Troy Lee**
Kolmogorov complexity and formula size lower bounds