# PROBABILISTIC TECHNIQUES IN STRUCTURAL COMPLEXITY THEORY

by

D. SIVAKUMAR

August 31, 1996

A dissertation submitted to the
Faculty of the Graduate School of
State University of New York at Buffalo
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

*Dedicated to S. Seshasayee*

## Acknowledgments

VELLORE, INDIA, 1981–1986.

The seeds for my zeal for higher studies, academic excellence and the teaching profession were sown during this period—my high school days. I am extremely grateful to my parents, to my brother and sister, and to numerous relatives for their constant encouragement that has been the foundation for all my later endeavors.

MADRAS, INDIA, 1986–1991.

My aunt and uncle deserve a special note of gratitude for their generous support during my stay in Madras. Their care and affection freed me from having to worry about numerous things and allowed me to focus on college. I thank my cousins for the wonderful time and all the fun that we had.

I am also thankful to the State and Central Governments for the low-cost high-quality education I was given at Anna University, Madras. I was very fortunate to find many wonderful friends at AU; my friends have remained a constant source of inspiration, support and encouragement through the years, and I am grateful for their friendship. Our lecturer at AU, Mr. S. Seshasayee, deserves my very special note of thanks for being my first inspiring (and inspired!) teacher at AU, and for encouraging me to go to the U.S. for graduate school. Without Seshu's encouragement, help and advice, I would have been writing systems software when I was writing my dissertation. I am thrilled to dedicate this dissertation to him.

BUFFALO, NY, USA, 1991–1996.

I am deeply grateful to my advisor Prof. Kenneth Regan for many different reasons. My first glimpse of theoretical Computer Science came with Ken's course and a subsequent independent study under his supervision during my first year at Buffalo. Ken's superb lectures and his terrific problem sets not only invited me to the exciting world of computational complexity theory, but also inspired me to work in this area. His continued nurturing was really key to my pursuit of research in complexity theory, and I am immensely grateful to him for this. During my early days in complexity theory, Ken generously provided me with bite-size problems to solve, and invited me to be his coauthor on my first papers in complexity theory [NRS94, CLL$^+$95]. The enthusiasm generated by the success of these endeavors has played a significant role in the later years. Ken has also been a patient sounding board for numerous half-baked ideas, some of which we successfully managed to turn into exciting theorems. During last month, Ken did another great help by reading drafts of this dissertation and providing very helpful comments in near-zero turnaround time. I thank him for all this and for his very friendly attitude towards me.

The arrival of Prof. Jin-Yi Cai in Buffalo was a major turning point in the course of my dissertation work in complexity theory. Much of the exciting work I have done in complexity theory was done jointly with him. His enthusiasm for research, his taste for problems, his inspiring lecturing style, and his relentless energy to obtain the strongest result possible are wonderful models to follow. Above all, I am very thankful to him for being very generous with his time, for being always available to discuss research or other matters. I owe much of my technical knowledge and my broad perspective on complexity theory to the numerous meetings I have had with Jin-Yi during the last two years.

I am grateful to Prof. Alan Selman for serving on my dissertation committee, and for providing numerous insightful comments during the course of my work. His comments have always been thought-provoking, and have shown the broader picture behind many a fine technical point.

It has been a great experience doing computational complexity theory at Buffalo. The numerous seminars and research meetings have been a great help in shaping my career as a complexity theorist and as an academic. The informal atmosphere that prevails at these meetings and the many many "what if" questions raised and answered have been simply wonderful. These meetings have also helped me to improve my presentation skills considerably. I thank Ken, Alan, and Jin-Yi for the superb organization of these seminars and research group meetings, and all the members of the Complexity Theory Research Group at Buffalo for their active participation and support. I also thank the complexity theory researchers at the University of Rochester—Lane Hemaspaandra, Mitsu Ogihara, Ioan Macarie, Marius Zimand and Gabriel Istrate—for the many wonderful UB–UR joint complexity seminars during the past three years. In fact, a good portion of the work in this dissertation owes its origins to a talk by Mitsu Ogihara at one of these seminars.

I thank Prof. Eric Allender of Rutgers University for serving as the external reader of my dissertation, and for his many helpful comments. He has also been very supportive of my research efforts, and I am grateful to him for that.

Besides Ken and Jin-Yi, I would like to thank Drs. Ashish Naik and Martin Strauss for their collaborations. Without these collaborations, this dissertation would not have succeeded. I also thank these people for allowing me to include much of our joint work in this dissertation.

I would like to express my gratitude to Professors Mitsunori Ogihara, Eric Allender, Jack Lutz and Steve Homer for their encouragement and support, as well as for their many valuable technical comments. I also thank Prof. Reuven Bar-Yehuda of Technion, Israel, whose visit to UB was a very enjoyable experience.

Four friends deserve special words of gratitude. Ashish Naik and Aravind Srinivasan have been sources of inspiration and guidance since my early years as a graduate student in complexity theory; I thank them for that and for their continued warm friendship. My sincere thanks to Ajay Nerurkar for numerous technical discussions and kicking around of ideas, and for being very quick in finding bugs in my various proofs of P = NP and of similar throems. I thank my long-time friend Ravikumar, now at Cornell University, for introducing me to the areas of program checking and self-testing, for his continued collaboration, and for the numerous telephone/email discussions we have had on complexity theory.

I thank the Department of Computer Science at Buffalo for the opportunity I was given; the Department of Computer Science and Professors Srihari and Regan for the financial assistantships during the last five years; and the University at Buffalo for a valuable fellowship. I thank Helene Kershner for the opportunities to teach various courses during my stay in Buffalo. I also thank the generous support of Professors Regan, Cai and Selman that made it possible to attend many conferences that were an integral part of graduate education.

The Department of CS at UB has been a great place for graduate education. I thank Elenora Heffner, Ellie Benzel, Gloria Koontz, Sally Elder, Jaynee Straw, and Joann Glinski for their friendly disposition and helpful nature in assisting with a mire of paperwork. I am thankful to Kannan Govindarajan, Vinay Dabholkar, Sridhar Seshadri, Sreenivas Gollapudi and Debra Burhans for taking an interest in and being supportive of my research endeavors.

The pressures of going through graduate school would have been torturous had it not been for the support of family and friends. First and foremost, I would like to thank my wife Uma Mahadevan for her constant support, patience, understanding, and encouragement. It was her love and affection that kept my sanity and cheer through the Fall of 1993 and the Spring of 1994 when my research appeared to be going nowhere. I am grateful to Uma for this and much more; without her solid support, this dissertation wouldn't exist.

I am happy to have had many friends in Buffalo, whose company I enjoyed very much. I thank Raman, Vinay, Gauri, Sreenivas, Rama, Kripa, Banu, Sridhar, Ranga, Sriganesh and Kannan for their friendship, and for all the fun we've had in the past few years. I thank my sister Veena and my brother-in-law Kannan for their help and support, and my long-time/long-distance friends Ancin, Badri, Balaji, Charu and Ravi for their continued friendship. I am happy to have lived in Buffalo, home of the Bills and neighbor of the Niagara Falls, and where the weather is always pleasant (except for some light flurries during a couple of winter months).

Ken Regan and Debbie Regan Howe (and Alex Regan) deserve a special note of thanks for their friendship, for the many invitations to their home, and for their very valuable advice and help in Childcare 101. Bharat Jayaraman has been a very friendly person who has taken continued interest in my progress as a graduate student; I am grateful to him and to Padma Jayaraman for their hospitality.

Finally, my love and affection go to two little ones: my nephew Anand, who I have watched from one day of age and whose progress it has been a great pleasure to follow; and the latest arrival, our son Aravind, in whose absence this dissertation would have been written in half the time, but whose smile was worth perhaps twice the delay.

# Contents

## Abstract

Randomization and derandomization are the two most significant twists in theoretical computer science during the last two decades. There are several problems for which the first efficient algorithms discovered were randomized, and several for which the first deterministic efficient algorithms were obtained by derandomizing the randomized algorithms. The two broad classes of tools used in the derandomization of randomized algorithms are *pseudorandom generators* and *quasirandom constructions*.

The central thesis of this work is that concepts, techniques, and ideas related to pseudorandomness and quasirandomness should find significant applications in the field of *structural complexity theory*. The structural approach to computational complexity is to classify problems into *complexity classes*, and to study various properties of the classes. The main contribution of this dissertation is to demonstrate the effectiveness of probabilistic techniques in two broad areas of structural complexity, namely *sparse hard sets* and *resource-bounded measure*.

We apply the randomization-derandomization paradigm to the question of whether sparse hard sets exist for the complexity classes P and NL. We prove that unless drastic collapses of complexity classes occur, there are no sparse hard sets for these classes, affirming two 1978 conjectures of J. Hartmanis. Our proof relies on algebraic structures related to the quasirandom construction used in the derandomization, and highlights the dramatic power of the probabilistic approach.

We establish connections between pseudorandom generators and the theory of resource-bounded measure, a complexity-theory analogue of classical measure theory. We show that if strong pseudorandom generators exist, then the class P/poly of languages with polynomial-size circuit families is not measurable within the class EXP of languages with exponential time algorithms. The class P/poly represents feasible problems under the Boolean circuit model of complexity, and is believed to be considerably weaker than the class EXP. Our result offers a new viewpoint on the P/poly vs. EXP question, and highlights the impact of pseudorandom generators to this question of central interest in complexity theory.

Finally, taking advantage of a known construction of a pseudorandom generator, we extend this connection to measurability within polynomial time, and prove several strong results that contribute to the still-evolving theme of resource-bounded measure at the polynomial time level.

# Chapter 1

# Introduction

The role of randomness in computation is perhaps the most important discovery in theoretical computer science during the last two decades. The excitement began in the late 1970's when some difficult problems, which had baffled computer scientists for many years, were found to possess very simple *randomized* algorithms. Algorithms that could figuratively toss a coin have since then been studied extensively, and this area of research has proved far more fruitful than researchers ever imagined. Four beautiful examples—primality testing, graph non-isomorphism, finding perfect matchings, and undirected graph connectivity—illustrate the power of randomness in four different contexts: respectively, sequential, nondeterministic, parallel, and space-bounded computation.

Given the seemingly ubiquitous power of randomness in computation, researchers have sought to understand exactly what power the coin toss adds to computation. On the theoretical front, it has always been clear that, similar to nondeterminism, randomness is a convenient *theoretical model* for certain types of computational problems and algorithms. However, despite the fact that the probability of error of most randomized algorithms can be made arbitrarily small, the inherent uncertainty is philosophically unpleasant to deal with. After all, a number is either a prime or it isn't one; of what use is it to say that it is almost certainly a prime? On the practical side, unlike nondeterminism, randomness has provided attractive options for *realistic efficient computation*—simpler and faster algorithms that are easier to program, near-certain guarantees about correctness, etc. Here, too, there is the dilemma that true randomness is not easily available to real computers, and one has to settle for deterministic rules (pseudorandom generators) that output a seemingly random sequence of bits.

To address such issues of theoretical significance as well as practical relevance, it is natural to study randomness as a *computational resource*, similar to time, space and nondeterminism. Of central importance in such a study are the questions:

(1) How much randomness is required for a given task?

(2) What property of a random sequence of bits makes a particular algorithm work?

(3) Is it possible to entirely dispense with randomness in a certain algorithm?

(4) What are the effects of using pseudorandom generators in lieu of true randomness?

A major success of the research during the past decade has been the formulation of these questions in precise mathematical terms. Building on these formalizations, considerable insight has been gained about the role of randomness in computation. The two most important concepts that have emerged, and that play a central role in this dissertation, are *pseudorandom generators* and *quasirandom constructions*.

Before introducing these notions in more precise terms, we pause to sketch a useful viewpoint of randomized computation that will naturally lead us to the definitions. Any randomized algorithm $A$ for a language $L$ can be thought of as a deterministic machine $D_A$, together with a sample space $\Omega$ and a distribution $\mu$ on $\Omega$. Given any input $x$, $A$ simply picks a random point $z$ from $\Omega$ according to the distribution $\mu$, and runs $D_A$ on the pair $(x,z)$. Depending on whether $D_A(x,z)$ produces the correct answer for the question "is $x \in L$?", the sample point $z$ can be classified as "good" or "bad." This defines a partition of $\Omega$ into a "good set" $G_x$ and a bad set $B_x$ of "witnesses." The celebrated "probabilistic method" contents itself with establishing that the good set $G_x$ is a large subset of $\Omega$ or, in some cases, with merely establishing that $G_x$ is non-empty! While this suffices for a non-constructive proof, it is very unsatisfactory from the viewpoint of complexity theory, where one is more interested in being able to *efficiently* and *deterministically* construct a good "witness" for every input $x$. Precisely how we achieve this gives rise to the two notions we alluded to; we will now describe them in more detail.

Informally speaking, a *pseudorandom generator* is an algorithm that stretches a short random string into a longer sequence of bits that "appear random" to machines with limited computational power. The philosophy of a pseudorandom generator is that how "random" a sequence of bits is depends on *who* is observing them. Suppose we know that the deterministic machine $D_A$ that uses a random string $z$ is very limited in its computational power—for instance, suppose there is a simple function $f$ that $D_A$ is incapable of computing. The idea is that we can perhaps "fool" the machine $D_A$ by providing it the result of applying the function $f$ to some *deterministically* chosen pieces of a short random string, often called the "random seed."

The main motivation for studying pseudorandom generation is that it is the most straightforward way to obtain a deterministic algorithm from a randomized algorithm: simply run through all possible seeds. Moreover, by its very definition, a pseudorandom generator is the most natural and general model in which to study issues about *complexity classes* as opposed to individual problems. This aspect of pseudorandom generators is very important in the study of structural complexity theory. Other motivations for studying pseudorandom generators come from cryptography and from practical considerations.

The concept of *quasirandom constructions* focuses on the inherent randomness content of a given set. Rather than generate outputs that "appear random" to a class of algorithms, the idea here is to to construct probability distributions that *approximate* the uniform distribution in certain specific and well-quantified ways. Seen another way, the goal is to construct a small sample space $\Theta$ that serves as a microcosm of a larger sample space $\Omega$; that is, the uniform distribution on $\Theta$ should possess properties similar, in some precisely-stated respects, to those of the uniform distribution on $\Omega$. For a quick example, consider the following randomized algorithm that attempts to find a large cut in a graph $G$: assign independently a random bit to each vertex, and partition the vertex set into those that receive a '0' and those that receive a '1'. It is clear that the expectation of the number of edges in the cut (edges across the partition) is precisely $|E(G)|/2$. A moment's reflection indicates that the analysis does not require the random-bit assignments to be fully independent, but only pairwise independent. Now we need only be able to construct a small sample space of 0-1 bit assignments that are pairwise independent—a much easier condition to meet.

Usually, the existence of small representative subsets of large sample spaces can be demonstrated by easy probabilistic arguments that offer no clue about how to construct the small sample space efficiently. In a very few cases, explicit and efficient constructions have been discovered, and these have proved very useful. A simple but amazingly powerful illustration of this is that of pairwise independence. Returning to our large-cut example, simple and elegant constructions of sample spaces of size $O(n)$ are known for generating $n$ pairwise independent 0-1 random variables [ABI86], in stark contrast to the sample space $\{0,1\}^n$ of size $2^n$.

Historically, research on pseudorandom generators has primarily been from a cryptographic viewpoint, and their usefulness is fairly well-understood in cryptographic settings. Quasirandom constructions have been studied extensively in coding theory and in combinatorics, and are beginning to find applications in theoretical computer science, specifically in the area of algorithm design.

## 1.1 This Dissertation

In this dissertation, we attempt to further the study of pseudorandom generators and quasirandom constructions from a *complexity-theoretic* viewpoint. The field of complexity theory, particularly structural complexity theory, has been a source of challenging problems that are mathematically very important. Unfortunately, almost all of the important questions in this field, such as $\mathrm{P} \overset{?}{=} \mathrm{NP}$ and $\mathrm{L} \overset{?}{=} \mathrm{P}$, remain open. Even more frustrating is the fact that many other questions, even ones that are presumably far simpler than these central problems, have also resisted traditional methods to a large extent. This situation has generated enormous interest and a need for applying new methods and tools. The central unifying idea of this dissertation is that concepts, techniques, and ideas involving pseudorandomness and quasirandomness will help us to gain better insight into complexity-theoretic questions. Furthermore, by posing new questions and problems, we hope that such a study will enrich our knowledge and understanding of pseudorandomness and quasirandomness.

ORGANIZATION OF THE DISSERTATION.

In Section 1.2, we set up the basic notations and definitions. In Section refprevious:work, we briefly outline some of the past work in the area of complexity theory that is concerned with pseudorandomness and quasirandomness.

In Chapter 2, we describe our work on *sparse sets*, a subject of fundamental interest in complexity theory. We consider two 20-year old conjectures of Hartmanis concerning the existence of sparse hard sets for the complexity classes P and NL. Our main results of this chapter affirm both conjectures. The proofs of our results in this chapter rely strongly on *randomization* and *derandomization*, particularly on the structure of a quasirandom construction. The proofs begin with a fairly simple probabilistic algorithm, which we first derandomize using a quasirandom construction. It turns out that the derandomized version of the probabilistic algorithm is not sufficient to settle the conjectures. Inspired by the algebraic structure of the quasirandom construction that we used, we introduce further twists from finite field theory to complete the proofs.

A very curious aspect of this chapter is how a certain twist that involves the quasirandom construction turns out to be the crucial step towards the resolution of the conjectures. Indeed, the proof techniques empolyed in this section lend ample testimony to our thesis that the newer probabilistic techniques invented in theoretical computer science should play a greater role in structural complexity theory. The main results reported in this chapter were obtained jointly with Dr. Jin-Yi Cai. Some of the related results we prove here were obtained jointly with Dr. Ashish Naik. These results appear in the papers [CS95a, CS95b, CNS96].

Chapter 3 describes our work on the impact of pseudorandom generators on the study of *resource-bounded measure theory*. This theory was developed by Dr. Jack Lutz as a generalization of classical measure theory, in order to apply measure-theoretic concepts to computational complexity theory. We consider the hypothesis that strong pseudorandom generators exist, and derive as a consequence that the class P/poly of languages with polynomial-size circuits does not have measure zero in exponential time. The measure of P/poly in exponential time has remained a rather mysterious and elusive question; our main result of this chapter sheds considerable light on this. Our result also shows that pseudorandom generators have significant impact on questions concerning strong separations between complexity classes.

The technical notion that inspired our proof is that of a *natural proof* [RR94], a recent breakthrough advancement in complexity theory. Our work in this chapter is also a step toward unifying this concept with resource-bounded measure theory, for we show that the two notions of resource-bounded measure and natural proofs are nearly the same. The research reported in chapter was conducted jointly with Drs. Kenneth Regan and Jin-Yi Cai, and is reported in [RSC95]. One of the technical lemmas used appears in [RS95].

In Chapter 4, we study the measure of the low-level circuit complexity classes $\mathrm{AC}^0$ and $\mathrm{AC}^0[\oplus]$. The results of this chapter are inspired by the results of Chapter 2, and take advantage of Nisan's strong pseudorandom generator [Nis91] secure against constant-depth circuits. We consider two different notions of measure on the complexity class P. Following the work in Chapter 2, we first prove a non-measure-zero result for the complexity class $\mathrm{AC}^0[\oplus]$ of languages accepted by polynomial-size, constant-depth circuits with AND, OR, NOT, and PARITY gates. Then we consider a different notion of measure and prove a pair of results that are fairly tight. On the one hand, we show that the class of languages accepted by constant-depth, nearly exponential-size circuits *does* have measure zero under this notion; on the other hand, we show that under the same notion of measure, $\mathrm{AC}^0[\oplus]$ does not have measure zero.

The results of Chapter 4 demonstrate more dramatic applications of pseudorandomness and quasirandomness as powerful tools in harnessing the full might of hardness results in complexity theory. We make extensive use of the pseudorandom generator against constant-depth circuits, and appeal to the various pseudorandom and quasirandom properties of the sample space produced by this generator. The results of this chapter were obtained jointly with Drs. Jin-Yi Cai and Martin Strauss, and appear in [CSS96]. One of the results is joint work with Dr. Kenneth Regan.

## 1.2 Preliminaries

We assume that the reader is familiar with the Turing machine model, and with the notion of an algorithm. We also assume that the reader is familiar with concepts such as nondeterministic Turing machines, encoding decision problems as languages over finite alphabets, etc. For more information on these, we refer the reader to standard textbooks such as [HU79] and [Pap94].

Throughout, we use the finite alphabet $\Sigma = \{0,1\}$; $\Sigma^*$ denotes the set of all possible strings of length zero or more over the alphabet $\Sigma$. We use the notations $\Sigma^{=n}$ and $\Sigma^n$ to denote the set of strings of length $n$ over $\Sigma$. We use $\Sigma^{\leq n}$ to denote the set of strings of length at most $n$ over $\Sigma$. Decision problems $D$ are encoded as languages $L_D \subseteq \{0,1\}^*$. For a language $L \subseteq \Sigma^*$, its *characteristic function* (also called *characteristic sequence*) $\chi_L : \Sigma^* \to \{0,1\}$ is defined by $\chi_L(x) = 1$ iff $x \in L$.

A Turing machine $M$ is said to be $t(n)$-time-bounded if for all $n$ and every $x \in \Sigma^n$, $M$ runs for no more than $t(n)$ steps. A Turing machine $M$ is said to be $s(n)$-space-bounded if for all $n$ and every $x \in \Sigma^n$, $M$ uses no more than $s(n)$ tape cells on its work tape. The complexity class DTIME$[t(n)]$ is defined to be the class of all languages over $\Sigma$ that are accepted by $t(n)$-time-bounded deterministic Turing machines. The complexity class NTIME$[t(n)]$ is defined to be the class of all languages over $\Sigma$ that are accepted by $t(n)$-time-bounded nondeterministic Turing machines. The complexity class DSPACE$[s(n)]$ is defined to be the class of all languages over $\Sigma$ that are accepted by $s(n)$-space-bounded deterministic Turing machines. The complexity class NSPACE$[s(n)]$ is defined to be the class of all languages over $\Sigma$ that are accepted by $s(n)$-space-bounded nondeterministic Turing machines.

The complexity class P is defined to be DTIME$[n^{O(1)}]$, NP is defined to be NTIME$[n^{O(1)}]$, and PSPACE is defined to be DSPACE$[n^{O(1)}]$. The exponential-time complexity classes E and EXP are defined, respectively, as DTIME$[2^{O(n)}]$ and DTIME$[2^{n^{O(1)}}]$. DSPACE$[O(\log n)]$ is often called Logspace or L, and its nondeterministic counterpart, NSPACE$[O(\log n)]$ is called Nondeterministic Logspace or NL. The quasipolynomial time complexity class QP is defined to be DTIME$[2^{(\log n)^{O(1)}}]$. The following relations exist among these classes: L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXP. Also, P $\subset$ QP $\subset$ E $\subset$ EXP.

A Boolean circuit with $n$ inputs and $m$ outputs is a directed acyclic graph with $n$ nodes of in-degree zero, called the *input nodes*, and one or more nodes with out-degree zero, called the *output nodes*. The interior nodes of a circuit are called *gates*; each gate computes a fixed (Boolean) function of the values on its in-edges, and places the result on its out-edge. The fixed Boolean functions computed by the gates include the Boolean AND, OR, and NOT functions. The function computed by the circuit is the function computed by its output gates, when the input values to the function are placed on the input nodes of the circuit. The *size* of a circuit is the sum of the number of nodes and the number of edges in the circuit; the *depth* of a circuit is the length of the longest path from any input node to any output node.

A family $\{C_n\}$ of circuits is said to accept a language $L$ if, for each $n$, $C_n$ is a circuit with $n$ inputs and one output such that for all $x \in \Sigma^n$, $C(x) = 1$ if and only if $x \in L$. The size and depth of the family of circuits $\{C_n\}$ are expressed as functions of $n$. A circuit family is said to be $\mathcal{C}$-uniform (for some complexity class $\mathcal{C}$ based on the Turing machine model) if there is a Turing machine $M$ of the type prescribed for $\mathcal{C}$ that for every $n$, outputs a description of the circuit $C_n$ when given $1^n$ as input. (Finer notions of uniformity exist, but are not relevant for the results of this dissertation.) When the uniformity of a circuit family is left unspecified, it is assumed that they are nonuniform, i.e. that there may be no Turing machine that produces the description of the circuits, or that the uniformity is irrelevant to the subject of discussion.

The class P/poly is defined to be the class of languages that are accepted by (nonuniform) families of Boolean circuits of polynomial size. The class $\mathrm{AC}^0$ is defined to be the class of languages that are accepted by (nonuniform) families of Boolean circuits of polynomial size and constant depth. The class $\mathrm{AC}^0[\oplus]$ is defined to be the class of languages that are accepted by (nonuniform) families of Boolean circuits of polynomial size and constant depth that are allowed to use PARITY gates in addition to AND, OR, and NOT gates. A PARITY gate, on inputs $y_1, y_2, \ldots, y_k$, outputs 1 iff the number of $y_i$'s that are 1 is odd. For $k \geq 0$, the class $\mathrm{NC}^k$ is defined to be the class of languages that are accepted by (nonuniform) families of Boolean circuits of polynomial size and $O(\log^k n)$ depth. The class NC is defined as the union of $\mathrm{NC}^k$ for $k \geq 0$.

One of the popular notions of uniformity used for definining the $\mathrm{NC}^k$ classes is that of *logspace uniformity*. A circuit family is said to be *logspace-uniform* if there is a $O(\log n)$-space-bounded Turing machine $M$ that for every $n$, outputs a description of the circuit $C_n$ when given $1^n$ as input. The class logspace-uniform $\mathrm{NC}^k$ is defined to be the class of languages that are accepted by logspace-uniform families of Boolean circuits of polynomial size and $O(\log^k n)$ depth. The following inclusion relationships exist among these classes: logspace-uniform $\mathrm{AC}^0 \subset$ logspace-uniform $\mathrm{AC}^0[\oplus] \subset$ logspace-uniform $\mathrm{NC}^1 \subseteq \mathrm{L} \subseteq$ logspace-uniform $\mathrm{NC}^2 \subseteq \ldots \subseteq$ logspace-uniform $\mathrm{NC}^k \subseteq \ldots \subseteq$ logspace-uniform $\mathrm{NC} \subseteq \mathrm{P}$.

For the purpose of defining complexity classes based on randomized algorithms, both Turing machines and Boolean circuits turn out to be convenient theoretical models. A probabilistic Turing machine is a Turing machine that has, besides input, output, and work tapes, an extra tape consisting of the outcomes of unbiased coin flips, called the random tape. At any point during its computation, the machine may consult the random tape, and make a probabilistic move in its execution. A probabilistic Boolean circuit has, in addition to its input nodes, additional nodes of in-degree zero that are fed the outcomes of unbiased coin flips.

A language $L$ is said to belong to the complexity class RP if there is a probabilistic polynomial-time bounded Turing machine $M$ such that for every $x$, $x \in L \Rightarrow \Pr[M \text{ accepts}] > 2/3$, and $x \notin L \Rightarrow \Pr[M \text{ accepts}] = 0$, where the probabilities are taken over the coin flips that define the contents of the random tape of the machine $M$. A language $L$ is said to belong to the complexity class co-RP if its complement belongs to RP. A language $L$ is said to belong to the complexity class BPP if there is a probabilistic polynomial-time bounded Turing machine $M$ such that for every $x$, $x \in L \Rightarrow \Pr[M \text{ accepts}] > 2/3$, and $x \notin L \Rightarrow \Pr[M \text{ accepts}] < 1/3$. The classes RP and co-RP are often called *one-sided error* probabilistic polynomial time classes, and BPP is called *two-sided error* probabilistic polynomial time. The complexity classes RL, co-RL, and BPL are defined analogously based on logspace-bounded Turing machines; in these cases, it is customary to make the restriction that the Turing machines have *one-way* access to their random tapes. The less standard model where they are allowed two-way access gives rise to the classes $\mathrm{RL}_2$, co-$\mathrm{RL}_2$, and $\mathrm{BPL}_2$.

The probabilistic circuit complexity classes $\mathrm{RNC}^1$, $\mathrm{RNC}^2$, etc. are also defined analogous to RP, based on the Boolean circuit model. Although logspace-uniform $\mathrm{NC}^1 \subseteq \mathrm{L}$, it is not known if logspace-uniform $\mathrm{RNC}^1 \subseteq \mathrm{RL}$. It is known, however, that logspace-uniform $\mathrm{RNC}^1 \subseteq \mathrm{RL}_2$.

The notions of *hardness* and *completeness* for complexity classes are important for the work described in this dissertation. First we define two notions of *reducibility*. A language $A$ is said to be *many-one reducible* to $B$ if there is a function $f : \Sigma^* \to \Sigma^*$ such that for all $x \in \Sigma^*$, $x \in A \iff f(x) \in B$. A language $A$ is said to be *Turing reducible* to $B$ if there is an oracle Turing machine $M$ such that $M$, using $B$ as an "oracle" (or as a "free" subroutine) accepts $A$. A language $B \in \mathcal{C}$ is said to be *hard* for

$\mathcal{C}$ under many-one reductions (resp. under Turing reductions) if every $A \in \mathcal{C}$ is many-one reducible (resp. Turing reducible) to $B$. A language $B \in \mathcal{C}$ is said to be *complete* for $\mathcal{C}$ under many-one reductions (resp. under Turing reductions) if $B \in \mathcal{C}$, and moreover, every $A \in \mathcal{C}$ is many-one reducible (resp. Turing reducible) to $B$. The complexity of computing the reductions is crucial, and is usually taken to be considerably weaker than what the complexity of languages in $\mathcal{C}$ is believed to be. The intuition is that a complete language $B$ is *archetypal* of the class $\mathcal{C}$ it represents, insofar as instances of all other languages in $\mathcal{C}$ can be recast as instances of $B$ with far fewer resources (time or space) than $\mathcal{C}$ itself talks about. For example, the standard notion of reducibility used in the context of NP-hardness or NP-completeness (see [GJ79], for example) is that of polynomial-time reducibility. For P-hardness or P-completeness, logspace-bounded reducibility or $\mathrm{NC}^1$ reducibility is commonly considered the standard. The most important complete problems for all the classes $\mathrm{L}, \mathrm{NL}, \mathrm{P}, \mathrm{NP}$, etc. are in fact complete under extremely weak reductions called *quantifier-free first-order projections*, ones that are even weaker than $\mathrm{AC}^0$ reductions.

One of the mathematical concepts that we use frequently in this dissertation is that of a finite field, for which we refer the reader to standard texts such as [LN86, Rom95]. All logarithms used in this dissertation are taken to the base 2.

## 1.3 A Brief Tour of some Gems in Complexity Theory

In this section, we briefly sketch previous work in complexity theory that concerns pseudo- and quasirandomness, and that has inspired our research. The choice of results included here is not meant to be comprehensive, but to give the reader a broad overview of the impact of randomness in complexity theory. The intent of this sketch is to give an account of the developments in chronological order, and to introduce the reader to some of the most important methods and tools that have been developed.

The connection between pseudorandom generators and complexity theory owes its origins to public-key cryptography, particularly to the seminal works of Diffie and Hellman [DH76] and Rivest, Shamir and Adleman [RSA78]. Shamir [Sha81] and Blum and Micali [BM84] were the first to make formal attempts to connect the notions of computational hardness and pseudorandomness, and to initiate a systematic study of complexity-based cryptography. Yao [Yao82] was the first to provide the field with the necessary formalism, and to prove the first significant results. The main contributions of Yao's work include a formal definition of the notion of a pseudorandom generator, and the demonstration that if certain problems are as hard as they are believed to be, then one can construct pseudorandom generators that are provably secure against attacks by feasible algorithms. Yao used this framework to show that if strong pseudorandom generators can be constructed, then polynomial time randomized algorithms (the class RP) can be deterministically simulated efficiently. Boppana and Hirschfeld [BH89] extended Yao's result to the class of randomized algorithms with two-sided error (the class BPP).

We digress at this point to mention another early and influential result, due to Sipser [Sip83]. This is also the first instance where a quasirandom construction was used effectively in complexity theory. Sipser showed that the class BPP is contained in the second level of the polynomial time hierarchy (PH). His proof used the concept of a *universal hash function* [CW79]. If $f$ is a truly random function from a set $A$ to a set $B$, then for any number of pairs $(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)$, where the $x_i$'s are distinct elements of $A$ and the $y_i$'s are arbitrary elements of $B$, the events "$f(x_i) = y_i$" are all independent of each other. Let $n = |A|, k = |B|$. Selecting a random function $f$ requires $n \log k$ random bits. A 2-universal hash function $h$, on the other hand, requires only $\log n \log k$ random bits to be specified, and guarantees that the events $h(x_i) = y_i$ are *pairwise independent*. Sipser's crucial observation was that the weaker requirement of pairwise independence was actually sufficient for the particular application. Note that when $n \approx k$, the number of random bits needed to specify a 2-universal hash function is polynomial in the number of bits needed to specify an element of $A$, whereas a truly random function requires an exponential number of bits. Another application of 2-universal hash functions was in the area of interactive computation. Goldwasser and Sipser [GS89] proved the surprising result that the secrecy of coin tosses did not add any power to a probabilistic polynomial time "verifier" who wishes to check the claim by an all-powerful "prover" that a string $x$ belongs to a language $L$.

We continue the discussion of pseudorandom generators. An interesting notion arising from complexity theory is that of a *one-way function*. Informally, a function is said to be one-way if it is easy to compute but very hard to invert. A classic example is that of multiplying two large integers. This function can be computed in deterministic quasilinear time, but the inverse function, namely integer factoring, is considered extremely hard and no efficient algorithm is known for this. Other examples are the discrete logarithm problem and the quadratic residuosity problem. The work of Yao and Blum and Micali focused on these hard problems. A central open question then was to know if it was possible to construct a pseudorandom generator based on *any* one-way function. Building on nearly a decade of research, Impagliazzo, Levin, Luby [ILL89] and Hastad [Hås90] showed that this task was indeed possible. Once again, the single most important idea used in their work was that of a 2-universal hash function. Suppose we have a random element $x$ from a set $S$ of $n$-bit strings, where $|S| \ll 2^n$. The entropy, or the randomness content, of the string $x$ is $\log|S|$ bits; however, the randomness is in an unusable form. The so-called *leftover hash lemma* of [ILL89] states roughly that if $h$ is a 2-universal hash function from $n$-bit strings to a set $A$ of size roughly equal to $S$, then the joint distribution of $\langle h, h(x) \rangle$ is very close to the uniform distribution. This is an example of the use of a quasirandom construction as a powerful tool that can "manipulate" entropy.

Around the same time, Nisan and Wigderson [NW88] studied the issue of designing pseudorandom generators based on an arbitrary hard problem, not necessarily the inverse of a one-way function. Through the use of certain (quasirandom-like) constructions called *combinatorial designs*, they provided an elegant construction of pseudorandom generators based upon very general complexity-theoretic hardness assumptions. Their work also gave an alternate proof of Sipser's result that BPP is included in the second-level of the polynomial-time hierarchy.

More recently, Razborov and Rudich [RR94] related the theory of pseudorandom generators to the *provability* of certain complexity class separations. They showed that if strong pseudorandom generators exist, then all known techniques, traditional and modern, of lower bounds are incapable of proving super-polynomial lower bounds on circuit size. We make essential use of their technology in the results of Chapter 3.

In the remainder of this chapter, we briefly outline some other applications of quasirandomness in complexity theory. Sipser [Sip88] showed that if certain quasirandom graphs called *expanders* exist, then either all randomized polynomial time algorithms can be derandomized, or non-trivial space-efficient simulations of time-bounded algorithms is possible. An expander is a constant-degree regular graph $G$ with a vertex set of size $N = 2^r$ that has many properties similar to a random graph. For example, a random walk of length $O(\text{diameter}(G))$ starting from a fixed vertex induces a near-uniform distribution on the vertex set $\{0,1\}^r$ of $G$. Such a walk uses $O(r)$ random bits and outputs $O(r)$ $r$-bit strings—a considerable savings from the obvious $O(r^2)$ random bits. Expander graphs have been studied extensively, and have proved to be a very valuable tool in computation theory. Ajtai, Komlos and Szemeredi [AKS87] showed how to (partially) derandomize space-bounded randomized algorithms by using expanders; Cohen and Wigderson [CW89] and Impagliazzo and Zuckerman [IZ89] showed how to amplify the success probability of randomized algorithms in a near-deterministic fashion (without having to repeat the algorithm with fresh random bits). Recently, Nisan and Zuckerman [NZ93] have improved the derandomization of space-bounded randomized algorithms. Their main result uses hash functions and limited independence of random variables, and shows that poly$(S)$ random bits do not add any power to space$(S)$ bounded machines!

Another complexity-theoretic pursuit that has made significant use of quasirandom constructions is the area of *probabilistically-checkable proofs* [AS92b, ALM$^+$92]. The breakthrough results of Arora et al. [ALM$^+$92] uses a combination of efficient quasirandom constructions—some of them studied earlier in coding theory—with limited independence. The essence of their result can be summarized by the following example. If $\varphi(x_1,\ldots,x_n)$ is a boolean formula in conjunctive normal form, one proof of the fact that $\varphi$ is satisfiable is to exhibit an assignment $\vec{a} = a_1a_2\ldots a_n$. This “proof” can be checked deterministically in polynomial time. The result of Arora et al. shows that it is possible to efficiently encode the assignment $\vec{a}$ so that by probing a *constant* number of bits in the encoding, one can verify, with high probability, if indeed $\varphi(\vec{a}) = \text{TRUE}$. This result has found numerous applications in proving the limitations on finding approximate solutions to hard combinatorial problems. Recently, Arora [Aro95] has shown the limitations of this approach in proving inapproximability; not coincidentally, his work is based on results about quasirandom constructions!

# Chapter 2

# Sparse Hard Sets

## 2.1 Background and Motivation

The P vs. NP question is arguably the most central open problem in complexity theory (in any event, it is the most popular one!). Since initial attempts to design polynomial time algorithms for NP-complete problems such as Satisfiability, Hamiltonicity, etc. failed miserably, researchers in the late 1960s turned towards attempts to prove P $\neq$ NP. The only weapon available in their arsenal with which to attack this problem was the technique of *diagonalization*. This technique had been very successful in proving the time-hierarchy and the space-hierarchy theorems, two cornerstones of the theory of computational complexity. Here, however, diagonalization didn't seem capable of accomplishing the task. A few years later, Baker, Gill, and Solovay [BGS75] demonstrated that it simply couldn't because it was too weak as a proof methodology: On the one hand, there exists a set $A$, which when used as an "oracle," makes deterministic polynomial time machines as powerful as their nondeterministic counterparts with the same oracle; on the other hand, any proof of P $\neq$ NP via diagonalization must necessarily carry over in the presence of any oracle.

The frustration that grew out of this and other attempts to separate P from NP led researchers to ask the following question: does the difficulty of separating P from NP stem from the choice of the Turing machine as the model of computation? In other words, is it the case that we are unable to focus on the essence of the matter because Turing machines draw our attention to the "wrong details," and are hard to analyze in a combinatorial manner? The "essence of the matter," of course, was believed to be the *combinatorial explosion* that occurs with NP-complete problems, both in terms of the large number and the wide variety of instances and in terms of the large solution space to search through for each instance. This belief resulted, in a somewhat counterintuitive fashion, in studying the P vs. NP question with a model of feasbile computation that is *stronger* than deterministic Turing machines. The most natural choice was to consider families of Boolean circuits.

Unlike a Turing machine, a single circuit cannot solve instances of a language of all lengths. Rather, the idea was to treat a language $L$ as feasible if there is a *family* of polynomial-size Boolean circuits $\{C_n\}$, such that the $n$-th circuit $C_n$ in the family will solve all instances of $L$ of length $n$. An important aspect of Boolean circuit complexity theory is that often one cares only about the *existence* of polynomial-size circuit families, not so much about how easy it is to *construct* such circuits. When one is concerned only with the existence of polynomial size circuits and not with the complexity of finding them, circuit complexity is often referred to as *non-uniform* complexity; the term *uniform* complexity is used for Turing machine complexity and for circuit complexity when one is also interested in the existence of a single algorithm (i.e. Turing machine) that generates the circuits for different input lengths.

The reason why polynomial-size circuits are stronger than polynomial-time Turing machines is two-fold. First, it is not hard to show that if a language $L$ can be decided by a polynomial time deterministic Turing machine, then there is a family of polynomial size circuits that decide $L$. In particular, this implies that if some NP-complete language can be shown not to have polynomial-size circuits, then it would follow that P $\neq$ NP. Secondly, this relationship is strictly one-way: any unary undecidable language can be encoded into polynomial-size circuits, but there is no Turing machine, polynomial-time or otherwise, that decides such a language.

The hope of the advocates of the Boolean circuit complexity viewpoint was that it would be easier to prove that a Boolean circuit of size $n^4$ cannot solve the Hamiltonian cycle problem on all $n$-vertex graphs. This hope was founded on the argument that an exponential number of all $n$-vertex graphs are Hamiltonian, and it would be impossible for a polynomial-size circuit to encode one tour for each Hamiltonian graph; nor would it be possible, the argument continues, for a small circuit to encode the Hamiltonian tours for all graphs in a compressed fashion, because of the wide variability among all Hamiltonian graphs. Thus, the argument concludes, a polynomial-size circuit family that decides Hamiltonicity simply *cannot exist*, much less producible in any uniform way. Thus the problem of proving that no Boolean circuit of polynomial size can solve all instances of the Hamiltonicity on $n$-vertex graphs is much more concrete than proving that no polynomial-time deterministic Turing machine can solve Hamiltonicity on all graphs, and hence hopefully more amenable to combinatorial techniques and non-constructive mathematics.

### 2.1.1 The Sparse Set Connection

By virtue of their definition, polynomial-size circuits are encodings of a "small amount" of information. However, to prove that NP-complete problems cannot be solved by polynomial-size circuits, presumably one still needs to address issues such as the types of gates of the circuit, its topology, and so forth. Thus, one might wonder: although the problem has been reduced to a concrete and combinatorial question, is it possible that the plethora of low-level details could still hamper progress on the question? (This was precisely the criticism leveled against Turing machines.) Is it possible to further distill the question to its real essence in purely mathematical terms? The answer is in the affirmative, and the key notions that achieve this are that of a *sparse set* and computational *reducibility*.

**Definition 2.1 (Sparse Set)** *A set $S \subseteq \Sigma^*$ is **sparse** if there exists a polynomial $p$ such that for all $n$, $|S \cap \Sigma^{\leq n}| \leq p(n)$.*

**Definition 2.2 (Turing reduction)** *A language $A$ is said to be **Turing reducible** to $B$ if there is an oracle Turing machine $M$ such that for all $x$,*

$$x \in A \iff M^B \text{ accepts } x.$$

*$A$ is said to be **polynomial-time Turing reducible** if $M$ runs in polynomial time.*

The connection between sparse sets and feasible computation by Boolean circuits is stated in the following easy, yet remarkably powerful, result (attributed to Albert Meyer in [BH77]).

**Proposition 2.1** *A language A can be decided by a family of polynomial-size Boolean circuits if and only if A is Turing reducible to a sparse set.*

Thus sparse sets serve as an important link in understanding the precise relationship between complexity classes defined under the Turing machine model and the Boolean circuit model. It also follows from Proposition 2.1 that to prove that NP-complete languages do not have polynomial size circuits, it is sufficient to show that they do not reduce to a sparse set via polynomial-time Turing reductions. Furthermore, this characterization also allows the study of *weaker* variants of this problem by placing restrictions on how the reduction may access the sparse set.

### 2.1.2 The Isomorphism Conjecture for NP

Historically, the main motivation for studying sparse sets comes from the famous paper of L. Berman and Hartmanis. Recall that one of the reasons for believing that P $\neq$ NP is the fact that NP-complete languages like Satisfiability and Hamiltonicity have exponentially many "YES" instances of each length, and moreover, the search space for each instance is too large for a polynomial time algorithm to search through. About the same time as the discovery of the connection between circuit complexity and sparse sets, Berman and Hartmanis [BH77] developed this observation on very solid technical grounds. The central notion in their work is that of an *isomorphism* between two languages, and the concept of *many-one reducibility*.

**Definition 2.3 (Isomorphism)** *Languages $A \subseteq \Sigma^*$ and $B \subseteq \Gamma^*$ are said to be* ***isomorphic*** *if there is a bijection $f : \Sigma^* \to \Gamma^*$ such that for any $x$, $x \in A$ iff $f(x) \in B$. A and B are said to be* ***polynomial time isomorphic*** *if the mapping $f$ and its inverse $f^{-1}$ are both computable in polynomial time.*

**Definition 2.4 (Many-one Reduction)** *A language A is said to be* ***many-one reducible*** *to a language B if there is a function $f$ such that for all $x$,*

$$x \in A \iff f(x) \in B.$$

*A is* ***polynomial time many-one reducible*** *to B if $f$ can be computed in polynomial time.*

Berman and Hartmanis proved that all the natural NP-complete languages (such as those found in [GJ79]) are isomorphic under polynomial-time computable bijections. Based on this evidence they conjectured that all languages complete for NP under polynomial time many-one reducibility are polynomial time isomorphic. The conjecture implies that there is essentially only one NP-complete language and that all the others are only cosmetically different.

Since the densities of any two polynomial time isomorphic sets are polynomially related, and since all known NP-complete sets are exponentially dense, their conjecture immediately implies that P $\neq$ NP, since any non-empty set is trivially complete for P under polynomial-time many-one reductions. Moreover, the same reasoning also implies that there can be no sparse complete set for

NP under polynomial time many-one reductions. Berman and Hartmanis went a step further, and conjectured that the above two consequences of the isomorphism conjecture are in fact equivalent. That is, there is no sparse complete set for NP under polynomial time many-one reductions unless P = NP.

Given the connection between reducibility to sparse sets and polynomial-size circuit families due to Proposition 2.1, the sparse set conjecture of Berman and Hartmanis can be thought of as a scaled-down version of the belief that NP-complete languages do not have polynomial-size circuit families. The sparse set conjecture states that if we restrict access to the sparse set to just one query that, in addition, has the property of a many-one reduction, then the sparse set is essentially useless to a polynomial-time computation.

The Berman-Hartmanis isomorphism conjecture has generated a lot of research in complexity theory. Building on earlier work by Fortune [For79], Mahaney [Mah82] showed that if NP has a sparse hard set (a weaker assumption than a sparse *complete* set) under polynomial time many-one reducibility, then P = NP. This is the definitive result concerning the nonexistence of sparse complete sets for NP under many-one reductions. Karp and Lipton [KL82] showed that if NP has a sparse hard set under Turing reductions, i.e. if NP has polynomial-size circuit families, then the polynomial hierarchy collapses to its second level, $\Sigma_2^p = \Pi_2^p$. In the subsequent years, considerable research effort has been devoted to studying various kinds of reductions to sparse sets, particularly aimed at generalizing Mahaney's result to weaker reducibilities. Particularly noteworthy is the result due to Ogihara and Watanabe concerning bounded truth table reductions of NP sets to sparse sets [OW91]; they showed that if the reduction to the sparse set is allowed only a constant number of queries, then it is essentially useless to a polynomial-time computation; i.e., if NP can be reduced to a sparse set with a constant number of queries, then P = NP.

### 2.1.3 The Conjectures of Hartmanis for P and NL

In 1978, Hartmanis [Har78] extended the work of Berman and Hartmanis to the case of P and nondeterministic logspace, NL. In each case, Hartmanis showed that all known complete problems for P and NL under many-one reductions computable in logspace (i.e. by a Turing machine that uses $O(\log n)$ space) are in fact isomorphic under bijections computable (and invertible) in logspace. Hartmanis made the analogous conjecture about the isomorphism of all P-complete and NL-complete languages under logspace many-one reductions. Furthermore, Hartmanis also conjectured that there are no sparse complete sets for P under logspace many-one reductions [Har78].

Unlike the sparse set conjecture for NP, there was little progress on the sparse set conjecture for P and NL during the last two decades. In 1994, Hemachandra, Ogihara, and Toda [HOT94] showed that if P has a *polylogarithmically* sparse hard set (only $O(\log^k n)$ strings up to each length $n$), then P = SC, the class of languages with algorithms that run simultaneously in polynomial time and polylogarithmic space. Recently, in early 1995, Ogihara [Ogi95] made a breakthrough development concerning this problem, and showed the following result:

**Theorem 2.2 ([Ogi95])** *If there is a sparse hard set for* P *under logspace many-one reductions, then* $\mathrm{P} \subseteq \mathrm{DSPACE}[\log^2 n]$.

The sparse set conjecture for NL remained completely open; the techniques of [HOT94] and

[Ogi95] were inadequate to attack this problem.

The research summarized in this chapter is concerned with the 1978 conjecture of Hartmanis about sparse hard sets for P and NL, and the related problem of sparse hard sets for P under randomized reductions. Our main results in this chapter, building on the work of Ogihara [Ogi95], affirm the sparse set conjectures of Hartmanis for P and NL.

**Theorem 2.3 (Main Theorem[CS95a])** *There is a sparse hard set for* P *under logspace computable many-one reductions if and only if* $\mathrm{P} = \mathrm{LOGSPACE}$.

**Theorem 2.4 (Main Theorem for** NL**[CS95a])** *There is a sparse hard set for* NL *under logspace computable many-one reductions if and only if* $\mathrm{NL} = \mathrm{LOGSPACE}$.

### 2.1.4 The Connection to Parallel Complexity

We mentioned Meyer's theorem (Proposition 2.1), which shows how sparse sets serve as a crucial link between the notions of feasible computation in the uniform and nonuniform complexity models. It turns out that the connection extends to the study of efficient parallel computation. It is another major open question in complexity theory whether every problem that has a feasible algorithm in fact has a highly parallelizable algorithm; i.e., whether $\mathrm{P} = \mathrm{NC}$. Recall that analogous to the two notions of feasible computation (uniform and nonuniform), we have two notions of efficient parallel computation defined in the uniform and nonuniform settings. The class (nonuniform) $\mathrm{NC}^1$ is the set of all languages that have a family of polynomial-size logarithmic depth circuits; uniform $\mathrm{NC}^1$ refers to the logspace-uniform counterpart. Proposition 2.5 below shows how sparse sets link the two notions.

IMPORTANT REMARK: Although there are finer notions of uniformity for the class $\mathrm{NC}^1$, the results of this chapter work only for logspace uniformity, so unless stated otherwise, "uniform" will mean "logspace-uniform," and plain $\mathrm{NC}^1$ will mean "nonuniform $\mathrm{NC}^1$."

**Definition 2.5 (**$\mathrm{NC}^1$ **reducbility[Coo85])** *A language $A$ is said to be* $\mathrm{NC}^1$ ***many-one reducible*** *to $B$ if there is a uniform family $\{C_n\}$ of polynomial-size logarithmic depth circuits with AND, OR, NOT, and ORACLE nodes such that for every $n$, $C_n$ accepts $A \cap \Sigma^n$ when the oracle is $B$. When counting the depth of the circuit $C_n$, each oracle node of fan-in $m$ contributes* $\log m$ *to the depth.*

An important consequence of this definition is that it preserves downward closure of $\mathrm{NC}^1$: if $B \in \mathrm{NC}^1$ and $A$ reduces to $B$ via $\mathrm{NC}^1$ reductions, then $A \in \mathrm{NC}^1$.

**Proposition 2.5** *A language $A$ has a family of polynomial-size logarithmic depth circuits, i.e. $A \in \mathrm{NC}^1$, if and only if $A$ reduces to a sparse set via uniform* $\mathrm{NC}^1$ *reductions.*

**Proof.** The proof from right to left is easy: Suppose $A$ reduces to a sparse set $S$ via the circuit family $\{C_n\}$ of size $s(n)$ and depth $d(n)$. By adding a look-up table for $S \cap \Sigma^{\leq s(n)}$ to the circuit $C_n$, and answering all oracle queries using this look-up table, we can solve $A \cap \Sigma^n$. The table is only polynomially large, so each look-up of $m$ bits takes only $O(\log m)$ depth, which is what the oracle node contributed to the depth originally. The size of the new circuit is also clearly polynomial in $n$.

For the converse, suppose $A$ has a family $\{C_n\}$ of polynomial-size logarithmic depth circuits. It is not hard to show that a polynomial-size logarithmic depth circuit may be converted to a polynomial-size logarithmic depth *formula*, i.e. to a circuit whose topology is that of a binary tree. Let $E(\cdot)$ denote a fixed function that encodes circuits such that $E(C_n)$ is a binary string whose length is at most a polynomial $e$ in the size of the circuit $C_n$. Define the set $S_A = \{1^n01^i0^{e(n)-i} \mid \text{ bit } i \text{ of } E(C_n) \text{ is } 1\}$. Clearly $S_A$ is sparse since each $E(C_n)$ is a string of length $n^{O(1)}$. The language

$$BFVP = \{\langle E,x\rangle \mid E \text{ is the encoding of a formula } C \text{ and } C(x) \text{ accepts } \}$$

was shown to be in NC$^1$ by Buss [Bus87, BCG$^+$92]. The circuit that reduces from $A$ to $S_A$, on input $x$, proceeds by first making all the queries $1^n01^i0^{e(n)-i}, 1 \le i \le e(n)$ in parallel to $S_A$, and then solves the *BFVP* problem on the instance $\langle E,x\rangle$, where $E$ is the sequence of bits produced as answers from the oracle $S_A$. This is clearly an NC$^1$ reduction. □

Below we state a stronger version of our main theorem. In the light of Proposition 2.5, this result can be viewed as shedding some light about the P vs. (nonuniform) NC$^1$ question, analogous to Mahaney's result vis-a-vis the NP vs. P/poly question.

**Theorem 2.6 (Stronger Version of Main Theorem[CS95a])** *There is a sparse hard set for* P *under many-one reductions computable in logspace-uniform* NC$^1$ *if and only if* P $=$ *logspace-uniform* NC$^1$.

An analogous strengthening applies to NL as well. This version of the main theorem assumes a stronger hypothesis and establishes a stronger consequence (since logspace-uniform NC$^1 \subseteq$ LOGSPACE); the reason we consider this to be a strengthening is because the resulting simulation of P is carried out more efficiently (in logspace-uniform NC$^1$).

### 2.1.5 The Circuit-Value Problem

The Circuit-Value Problem, abbreviated *CVP*, consists of pairs $\langle C,x\rangle$ where $C$ is a Boolean circuit with $k$ inputs, $x \in \{0,1\}^k$, and on input $x$, $C$ outputs '1.' Ladner [Lad75] showed that *CVP* is complete for P under logspace many-one reductions, and Cook [Coo85] observed that *CVP* is, in fact, complete for P under NC$^1$ reductions. In fact, Immerman [Imm87] has shown that *CVP* is complete for P under an extremely weak form of reducibility using the notion of a quantifier-free first-order projection.

### 2.1.6 Overview of the Proof Technique

We prove our main theorem in three successive stages. In each stage, we assume that there is a sparse hard set for P under logspace (or NC$^1$) many-one reductions, and show how to solve the P-complete Boolean Circuit Value Problem (*CVP*) in a space-efficient or highly parallel fashion. The successive solutions for *CVP* become more and more efficient. In the first stage, we show how to solve *CVP* in logspace-uniform randomized $\mathrm{NC}^2$; in the second stage we improve this solution to run in logspace-uniform deterministic $\mathrm{NC}^2$; and in the last stage we show how to solve *CVP* in logspace-uniform $\mathrm{NC}^1$. The first two stages yield results that fall short of settling the sparse set conjecture for P, and it is the third stage where the strongest form of our theorem is proved. In fact, the proof of the main theorem could have been described entirely independently of the first two stages. However, we prefer to describe the result in three stages because the final proof of the main theorem uses many ideas whose roots can be traced back to the weaker results proved in the first two stages.

The main ingredient in the resulting simulation of P in each stage is the solution of a system of linear equations over a finite field. The first stage begins with a crucial observation due to Ogihara [Ogi95]. Based on this observation, we employ a probabilistic process that sets up a system of linear equations over the field $\mathbb{Z}_2$. Then we prove a probabilistic lemma that guarantees that the system of equations has sufficiently high rank. By solving this system of equations, we obtain an $\mathrm{RNC}^2$ simulation of P. In the second stage, we use a "small-bias sample space" construction [NN93, AGHP92] to derandomize this algorithm to obtain an $\mathrm{NC}^2$ simulation. In the third stage, exploiting additional algebraic properties of a closely related construction, we arrive at a Vandermonde system. We then solve the system using a closed formula that involves the elementary symmetric polynomials over a certain finite field. The key technical idea to implement the formula in $\mathrm{NC}^1$ is the use of the discrete Fourier transform. The final result is a collapse of P to logspace uniform $\mathrm{NC}^1$.

To settle the sparse set conjecture of Hartmanis for NL, additional ideas will be required; we describe them in Section 2.6.

## 2.2 The First Stage: an $\mathrm{RNC}^2$ Simulation

In this section, we consider the hypothesis that there is a polynomially sparse set $S$ hard for P under logspace (or even $\mathrm{NC}^2$) many-one reductions. Note that the sparse set $S$ need not belong to P itself. (Thus our assumption is even weaker than P-*completeness* as stated in the conjecture of Hartmanis.) The framework and basic ideas introduced here are used throughout this chapter.

### 2.2.1 The Strategy: Taking Advantage of Sparseness

The strategy for the proof will be the following. To solve an instance $\langle C, x\rangle$ of *CVP*, it is clearly sufficient to compute the value of the output gate of $C$ on input $x$. Suppose we invoke the hypothesis and reduce this instance to an instance $w$ of the sparse set $S$ so that $C(x) = 1$ iff $w \in S$. While it is true that $S$ has only polynomially many strings of length $|w|$, we still seem to have made no progress, since we have no way of enumerating those strings, nor do we know how to efficiently check if $w \in S$. However, let us pause for a moment to recall the reason why we believe sparse hard sets are unlikely: it is precisely the fact that they can't efficiently encode the solutions to exponentially many different instances of a language. To turn this to our advantage, we consider the following question: is there a way to produce exponentially many different questions about the behavior of $C$ on $x$, all of which can then be rephrased as a query to the sparse set $S$? The rub, however, is that $C(x)$ is just one bit, and how are we to ask exponentially many different questions about it? Again, the problem hints at the solution: what if we set for ourselves the more ambitious goal of finding the value of *every* gate of $C$ on input $x$? If $C$ has $n$ gates, then our goal is to find the $n$-bit string $g = g(C,x) \in \{0,1\}^n$ that gives the value of each gate of $C$ on input $x$. Now, of course, there is a natural way of producing $2^n$ questions about $C(x)$: if we treat $g$ as an $n$-variable linear function over the field $\mathbb{Z}_2$ defined by

$$g(Z_1, Z_2, \ldots, Z_n) \doteq (\sum_i g_i Z_i) \bmod 2,$$

then we may ask $2^n$ different queries about $g$. The next issue is whether these questions about the string $g$ can be rephrased as queries to the sparse set $S$. To this end, we may define an appropriate language $A$ which consists of instances precisely of this form: a circuit $C$, an input $x$, some $Z \in \mathbb{Z}_2^n$, and a bit $b \in \mathbb{Z}_2$, so that the tuple $\langle C, x, Z, b\rangle \in A$ iff $g(Z) = b$, where $g = g(C, x)$. It is clear that $A \in$ P; hence by hypothesis, $A$ many-one reduces to $S$. Let $f$ be a logspace computable function such that for all $x$, $x \in A \iff f(x) \in S$. In the remainder of this section, we will show how to solve *CVP* in RNC$^2$ using the reduction $f$ from $A$ to $S$.

Continuing the line of thought initiated above, let us ask what happens if we reduce all possible instances $\langle C, x, Z, b\rangle$ to the sparse set $S$. For any $C, x, Z$, exactly one of the values $b = 0, 1$ satisfies the equation, and thus exactly one of $f(\langle C, x, Z, 0\rangle)$ and $f(\langle C, x, Z, 1\rangle)$ is a string in $S$. Moreover, suppose that for two *distinct* vectors $Z$ and $Y$ and some pair of values $b, b' \in \mathbb{Z}_2$, $f(\langle C, x, Z, b\rangle) = f(\langle C, x, Y, b'\rangle) = w$. When this happens, $g(Z) = b \iff w \in S \iff g(Y) = b'$, which implies that $g(Z) = b \iff g(Y) = b'$. In other words, regardless of whether the image $w$ is in $S$ or not, i.e. regardless of whether $g(Z) = b$ and $g(Y) = b'$ are true or not, they hold or fail simultaneously. Since $g$ is a linear function, $(g(Z) = b \iff g(Y) = b')$ is true iff $g(Z + Y) = b + b'$, and we have an equation mod 2 on the values of the gates of $C$ on input $x$. Note that the additions $Z + Y$ and $b + b'$ are performed mod 2. Since $Z \neq Y, wehave Z + Y \neq 0^n$, and thus we have a non-trivial equation over $\mathbb{Z}_2$ on the variables $g$.

Fix any $C$ and $x$, and let $n$ denote the number of nodes in $C$ (including the inputs, output, and the interior gates). Let $N$ denote the largest value of $|f(\langle C, x, Z, b\rangle)|$ (over all $Z$ and $b$). Clearly $N$ is polynomially bounded in $n$. Let $p(n)$ be a polynomial function that bounds the number of strings in $S$ of length at most $N$. Suppose for the moment that we produce all the queries $f(\langle C, x, Z, b\rangle)$. Each $Z$ gives rise to exactly one query that must be a member of $S$. Since there are only $p(n)$ many strings in $S$, some string $w \in S$ must be the image of at least $2^n/p(n)$ many tuples $\langle C, x, Z, b_Z\rangle$, where $b_Z$ is the correct value of $g(Z)$. This is precisely where the sparseness of $S$ kicks in, and this is precisely what we plan to take advantage of.

The definition of the language $A$ and the observation about the existence of a "popular" string $w \in S$ are due to Ogihara [Ogi95]. At this point, Ogihara used a deterministic rule to identify a collection of $2^{O(\log^2 n)}$ vectors $Z$ and proceeded to compute the gate values of $C$. Instead we introduce our first idea here: *randomization.*

For notational simplicity we assume $p(n)$ is a power of 2; in particular, we will assume that $\log_2 p(n)$ is always an integer. Our idea is to randomly and uniformly choose *polynomially* many vectors $Z \in \mathbb{Z}_2^n$ and compute both $f(\langle C, x, Z, 0\rangle)$ and $f(\langle C, x, Z, 1\rangle)$, collecting an equation whenever a "collision" takes place. We remind the reader once again that whenever $f(\langle C, x, Z, b\rangle)$ and $f(\langle C, x, Y, b'\rangle)$ collide for $Z \neq Y$, irrespective of whether the image is a member of $S$ or not, we can produce a valid equation.

The next question is: does the system of equations thus produced have sufficiently high rank, so that we may solve them to infer $g$? The following lemma ensures that this process gives us a system of linear equations of rank $n - O(\log n)$, even if we restrict attention to collisions that take place on a single popular $w \in S$.

### 2.2.2 A Probabilistic Lemma

Let $B = \{0,1\}^n$ denote the $n$-dimensional binary cube. With respect to the finite field of two elements $GF(2) = \mathbb{Z}_2$, $B$ is a vector space of dimension $n$. Let $T \subseteq B$ be an arbitrary subset of the cube. We ask the following question: If we uniformly and independently pick a sequence of $m$ points in $B$, what can we say about the probability distribution of the dimension of the affine span of those points picked from $T$ as a function of $m$, $n$ and $|T|$?

**Lemma 2.7** *Let $k$ be a power of 2. If $|T| \geq 2^n/k$, where $k = n^{O(1)}$, then for $m = 2kn^2 + n = n^{O(1)}$, if we uniformly and independently pick a sequence of $m$ points in $B$, the probability that the dimension of the affine span of the points chosen from $T$ is less than $n - \log_2 k$ is at most $e^{-n^2+O(n\log n)}$.*

**Proof.** Consider any sequence of points of $B$ being picked by the above process. Let us mark any such sequence $p_1, p_2, \ldots, p_m$ by a 0-1 sequence of the same length $m$ according to the following rule: Suppose the subsequence $p_{i_1}, p_{i_2}, \ldots, p_{i_\ell}$ is the intersection of the sequence $\{p_i\}$ with the set $T$. Mark $p_{i_1}$ with a 0. For $j > 1$, a point $p_{i_j}$ is marked 1 iff the dimension of the affine span of $p_{i_1}, p_{i_2}, \ldots, p_{i_j}$ is greater than that of $p_{i_1}, p_{i_2}, \ldots, p_{i_{j-1}}$. All other points in $\{p_i\}$ are marked 0. This defines a 0-1 sequence $\sigma$ of length $m$. We wish to estimate the probability that the number of 1's in $\sigma$ is small.

The process of uniformly and independently picking a sequence of $m$ points in $B$ induces a probability distribution over the set of 0-1 sequences $\sigma$ of length $m$ defined as above. Suppose we have picked a sequence $p_1, p_2, \ldots, p_{i-1}$ that intersects with $T$ in a set whose affine span has dimension $< n - \log_2 k$. Then there are at least $|T| - 2^{n-\log_2 k-1}$ points of $T$, which, if picked next, would increase the dimension of the affine span of the intersection. This cardinality is $\geq 2^n/k - 2^n/(2k) = 2^n/(2k)$. Hence the conditional probability of the next bit being 1 satisfies the bound

$$\Pr[\sigma_i = 1 \mid \text{ the number of 1's in } \sigma_1, \ldots, \sigma_{i-1} < n - \log_2 k] \geq 1/(2k).$$

Then for any sequence $\sigma$ with strictly fewer than $n - \log_2 k$ many 1's,

$$\Pr[\sigma] \leq \left(1 - \frac{1}{2k}\right)^{m-(n-\log_2 k)},$$

which is bounded above by $e^{-n^2}$ if $m = 2kn^2 + n$. Therefore,

$$\Pr[\dim(\text{affine span of } \{p_i\}_{i=1}^m \cap T) < n - \log_2 k] \leq \sum_{j<n-\log_2 k} \binom{m}{j} e^{-n^2} < e^{-n^2+O(n\log n)}.$$

□

### 2.2.3 Wrapping it up: Parallel Solution of Linear Equations

By the above lemma, if we uniformly and independently choose polynomially many $Z$'s and produce the queries $f(\langle C,x,Z,b\rangle)$ for $b = 0,1$, collecting equations whenever possible, with high probability we will obtain a system of linear equations with rank deficiency at most $\log_2 p(n)$. It is clear that the process of creating the instances, making the queries, and collecting the equations may be accomplished by a polynomial-size circuit of logarithmic depth (not counting the depth to simulate the reduction $f$). Since $f$ is assumed to be computable in logspace (or in $\mathrm{NC}^2$), the whole process, including the reduction, may be accomplished in depth $O(\log^2 n)$. We now describe how we can use the equations to determine in $\mathrm{NC}^2$ the outputs of all the gates of $C$ on input $x$.

Without loss of generality, let the rank of the system be $n - \log_2 p(n)$, and let $m(= n^{O(1)})$ denote the number of equations we have. Denote the equations by $E_1,\ldots,E_m$, and for $i \geq 1$, call an equation $E_i$ *useful* if the rank $rk(E_1,\ldots,E_i) > rk(E_1,\ldots,E_{i-1})$. Clearly the number of useful equations is at least $n - \log_2 p(n)$; without loss of generality, we will assume that we have exactly $n - \log_2 p(n)$ useful equations. Mulmuley [Mul87] gives an algorithm to compute the rank of an $\ell \times n$ matrix, which, for $\ell = n^{O(1)}$, can be implemented by a circuit of depth $O(\log^2 n)$ and size $n^{O(1)}$. For $1 \leq i \leq m$, we compute in parallel $rk(E_1,\ldots,E_i)$, and identify all the useful equations. Now we have $n - \log_2 p(n)$ equations in $n$ variables, with rank $n - \log_2 p(n)$. We apply the same process to the columns, and identify the $(n - \log_2 p(n))$-many useful columns. We rename the variables so that the first $n - \log_2 p(n)$ columns are all useful.

Finally, in parallel we create, for each of the $p(n)$ possible assignments to the last $\log_2 p(n)$ variables, a system of $n - \log_2 p(n)$ equations as an $(n - \log_2 p(n)) \times (n - \log_2 p(n))$ matrix. Each one of these can be solved in $\log^2 n$ depth and $n^{O(1)}$ size using the algorithm due to Borodin, et al. [BvzGH82]. Exactly one of the $p(n)$ systems gives the correct values of the gates of the circuit $C$ on input $x$. To identify the correct solution, we can check (again in parallel) the validity of each solution using the local information about the circuit $C$ and input $x$ such as $x_i = 0$ or $x_i = 1$, or $g_j = g_k \wedge g_\ell$, etc. There will be a unique solution that passes all such tests and, in particular, we will find the correct output of $C(x)$. Clearly this checking process does not require more than depth $O(\log n)$. Since the only possibility of error in this process is not producing a system of equations of sufficiently high rank. We have proved:

**Theorem 2.8** *If there is a sparse set that is hard for* P *under logspace or* $\mathrm{NC}^2$ *many-one reductions, then* $\mathrm{P} \subseteq \mathrm{RNC}^2$.

## 2.3 Stage Two: An $\mathrm{NC}^2$ Solution via Derandomization

In this section, we use a small sample space construction due to Alon et al. [AGHP92], and generalize their result concerning the construction. We apply the generalization to derandomize the probabilistic simulation of Section 2.2. Under the hypothesis about sparse hard sets, this yields a collapse of P to $\mathrm{NC}^2$.

We first attempt to address the question: what does it take to derandomize the algorithm of the previous section? Fix an arbitrary instance $\langle C,x\rangle$ of *CVP*. As before we have $B = \{0,1\}^n = \mathbb{Z}_2^n$ considered as an $n$-dimensional vector space over the finite field $\mathbb{Z}_2$. For each $Z \in B$, let $b_Z = g(Z)$

be the correct value of the linear function $g$ defined by the sequence of values of the gates of the circuit $C$ on input $x$. Let us call the string $w = f(\langle C, x, Z, b_Z \rangle) \in S$ the *color* of $Z$. Thus the reduction to the sparse set $S$ gives a coloring of $B$ with at most $p(n)$ colors. Let $D$ be a subset of $B$ of cardinality bounded by a certain polynomial in $n$. The coloring of $B$ induces a coloring of $D$; thus $D$ is the union of at most $p(n)$ many color classes:

$$D = C_1 \cup C_2 \cup \ldots \cup C_{p(n)}.$$

Let the affine span of $C_i$ be denoted by $L_i + d_i$, where $L_i$ is a linear subspace, and $d_i$ is a displacement vector. Let $L = L(D) = L_1 + L_2 + \ldots + L_{p(n)}$ be the sum of the linear subspaces. We call $L$ the *span* of the color classes. $L_i$ is spanned by differences of vectors in $C_i$. For some spanning set of vectors of $L_i$, each vector in the set gives us an equation mod 2 of the values of the gates of $C$ on input $x$. If we collect a generating set of vectors for each $L_i$, together they span $L$. Thus, if we can construct a set $D$ with polynomial size and with $\dim L(D) \geq n - O(\log n)$ irrespective of how $D$ is colored, we will have succeeded in derandomizing the construction of the last section. That is, by sampling exhaustively in $D$, we will have obtained a system of linear equations of rank $\geq n - O(\log n)$.

We claim that the above task can be accomplished as follows: given $p(n)$, construct a polynomial sized set $D$ such that for any linear subspace $M$ of $B$ with $\dim M < n - \log_2 p(n)$, and any $p(n)$ displacement vectors $b_1, \ldots, b_{p(n)} \in B$, the union of the $p(n)$ affine subspaces $\bigcup_{i=1}^{p(n)} (M + b_i)$ does not cover the set $D$. For if so, then no matter what the induced coloring on $D$ is, the span of the color classes $L$ must be of dimension $\geq n - \log_2 p(n)$, simply because the union of at most $p(n)$ affine subspaces $\bigcup_{i=1}^{p(n)} (L + d_i)$ does cover $D$:

$$\bigcup_{i=1}^{p(n)} (L + d_i) \supseteq \bigcup_{i=1}^{p(n)} (L_i + d_i) \supseteq D.$$

Let $k = 1 + \log_2 p(n) = O(\log n)$. Without loss of generality, we may assume such a linear subspace $M$ has dimension exactly $= n - k$. Any such $M$ can be specified as the null space of a system of linear equations

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = 0,$$

where $i = 1, \ldots, k$, and the $k$ vectors $\{(a_{i1}, a_{i2}, \ldots a_{in}) \mid i = 1, \ldots, k\}$ are linearly independent vectors in $B$ over $\mathbb{Z}_2$.

### 2.3.1 Construction of the Sample Space

Let $m = 2k + \log_2 n + 1 = 2\log_2 p(n) + \log_2 n + 3 = O(\log n)$. The Galois field $\mathbb{F} = GF(2^m)$ has a vector space structure over $GF(2)$ of dimension $m$. Choose any basis $\{e_1, \ldots, e_m\}$, then for $u = \sum_{i=1}^m u_i e_i$ and $v = \sum_{i=1}^m v_i e_i$ in $\mathbb{F}$, we can define an inner product by letting

$$\langle u, v \rangle = \sum_{i=1}^{m} u_i v_i,$$

and doing all arithmetic over $\mathbb{Z}_2$.

The (multi)set $D$ is a "small-bias sample space" [AGHP92], defined as follows:

$$D = \{(\langle 1,v\rangle, \langle u,v\rangle, \ldots, \langle u^{n-1},v\rangle) \mid u,v \in \mathbb{F}\}.$$

Note that $|D| = 2^{2m} = n^{O(1)}$.

Consider any non-zero vector $a = (a_0, a_1, \ldots, a_{n-1}) \in B$ and any $b \in \mathbb{Z}_2$. We wish to estimate the size of the intersection of $D$ with the affine hyperplane $\sum_{i=0}^{n-1} a_i x_i = b$.

Since the inner product $\langle \cdot,\cdot \rangle$ is bilinear over $\mathbb{Z}_2$ we have

$$\sum_{i=0}^{n-1} a_i \langle u^i, v\rangle = \langle \sum_{i=0}^{n-1} a_i u^i, v\rangle.$$

Let $q_a(X)$ denote the polynomial $\sum_{i=0}^{n-1} a_i X^i \in \mathbb{F}[X]$. If $u$ is a root of the polynomial $q_a(X)$, then clearly the inner product $\langle \sum_{i=0}^{n-1} a_i u^i, v\rangle = 0$. Now suppose $u \in \mathbb{F}$ is not a root of $q_a(X)$, then $\sum_{i=0}^{n-1} a_i u^i = q_a(u)$ is a non-zero element in $\mathbb{F}$. It is easy to see that for any non-zero $w \in \mathbb{F}$,

$$\Pr_{v\in\mathbb{F}}[\langle w,v\rangle = 0] = 1/2.$$

Thus,

$$\begin{aligned} &\Pr_{u,v\in\mathbb{F}}[\sum_{i=0}^{n-1} a_i\langle u^i,v\rangle = 0] \\ = \ &\Pr_{u\in\mathbb{F}}[u \text{ is a root of } q_a(X)] + \Pr_{u\in\mathbb{F}}[u \text{ is not a root of } q_a(X)] \cdot 1/2. \end{aligned}$$

But $q_a(X)$ is a non-zero polynomial of degree at most $n-1$, thus

$$\Pr_{u\in\mathbb{F}}[u \text{ is a root of } q_a(X)] \le \frac{n-1}{2^m}.$$

Collecting terms, we have

$$\Pr_{u,v\in\mathbb{F}}[\sum_{i=0}^{n-1} a_i\langle u^i,v\rangle = 0] \le \frac{1}{2} + \frac{n-1}{2^{m+1}}.$$

In particular, if $m > \log_2 n$, both affine hyperplanes $\sum_{i=0}^{n-1} a_i x_i = 0,1$ must intersect our set $D$. The bound above was shown in [AGHP92]. We strengthen it to handle $O(\log n)$ linearly independent equations, not just these two hyperplanes.

In general, consider any $k$ linearly independent equations $\sum_{j=0}^{n-1} a_{ij} x_j = b_i$, where $a_{ij}, b_i \in \mathbb{Z}_2$, and $i = 1,\ldots,k$. Denote this affine space by $\Pi$. Denote the point in $D$ specified by $u,v$ as $D(u,v)$. We wish to estimate the probability $\Pr_{u,v\in\mathbb{F}}[D(u,v) \in \Pi]$.

Let $Q$ denote the following set of polynomials: $\{\sum_{i=1}^{k} \beta_i[\sum_{j=0}^{n-1} a_{ij}X^j] \mid \beta_i \in \mathbb{Z}_2, \text{ not all } \beta_i \text{ zero}\}$. We claim that the cardinality of $Q$ is exactly $2^k - 1$, and none of the polynomials in $Q$ is the zero polynomial. This follows from the fact that the vectors $(a_{i0}, \ldots, a_{i,n-1})$ are linearly independent over $\mathbb{Z}_2$. Consider the system of linear equations on the $m$ bits of $v$:

$$\sum_{j=0}^{n-1} a_{ij}\langle u^i, v\rangle = \left\langle \sum_{i=0}^{n-1} a_{ij} u^i, v \right\rangle = b_i, \qquad i = 1,\ldots,k.$$

We claim that if $u \in \mathbb{F}$ is not a root of any polynomial in $Q$, then the coefficient vectors of this system of equations are linearly independent over $\mathbb{Z}_2$. For otherwise, a non-zero linear combination of the coefficient vectors of $v$ will be zero, which is precisely the same as $u$ being a root of one of the polynomials in $Q$. Thus, the conditional probability for $v$ to satisfy this linear equation system is precisely $1/2^k$. However, since $|Q| = 2^k - 1$, and each polynomial in $Q$ is non-zero and of degree at most $n-1$,

$$\Pr_{u \in \mathbb{F}}[u \text{ is a root of some polynomial in } Q] \leq (2^k-1)(n-1)/2^m.$$

Collecting terms, we obtain

$$\begin{aligned} & \left| \Pr_{u,v \in \mathbb{F}}[D(u,v) \in \Pi] - \frac{1}{2^k} \right| \\ \leq \quad & \frac{(2^k-1)(n-1)}{2^m} \left(1 - \frac{1}{2^k}\right) \\ < \quad & \frac{n}{2^{m-k}}, \end{aligned}$$

which by our choice of $m$ and $k$ is bounded above by $1/2^{k+1}$. Thus, in particular,

$$\Pr_{u,v \in \mathbb{F}}[D(u,v) \in \Pi] > 0.$$

Other than linear independence, the coefficient vectors and the right hand side vector $b_1, \ldots, b_k$ in the definition of $\Pi$ are arbitrary; the total number of the $b$ vectors is $2^k = 2p(n) > p(n)$, and it follows that no linear subspace $M$ of dimension $< n - \log_2 p(n)$ can cover the set $D$ with some $p(n)$ displacements. It follows that if we sample exhaustively from $D$, the resulting system of equations will have rank $\geq n - \log_2 p(n)$. With these we can solve *CVP* on the instance $\langle C, x \rangle$ just as before in $\mathrm{NC}^2$, using the algorithm due to Borodin, et al. [BvzGH82]. We have proved:

**Theorem 2.9** *If there is a sparse set $S$ that is hard for* P *under* $\mathrm{NC}^2$ *many-one reductions, then* $\mathrm{P} = \mathrm{NC}^2$.

## 2.4 The Finale: $\mathrm{NC}^1$ Simulation

The collapse of $\mathrm{P} = \mathrm{NC}^2$ under the assumption about sparse sets does not suffice for our ultimate goal of settling the sparse set conjecture for P. In this section, we build on ideas from the previous sections, and show that if there is a sparse set $S$ that is hard for P under many-one reductions computable in logspace, then P = LOGSPACE. In fact, we prove the following stronger statement:

**Theorem 2.10** *If a sparse set $S$ is hard for* P *under any kind of many-one reductions, then the* P*-complete circuit-value problem can be solved by a logspace-uniform family of polynomial size, logarithmic depth circuits that make polynomially many parallel calls to the reduction.*

That is, modulo the complexity of the reduction to the sparse set, the resulting algorithm can be implemented by a uniform circuit of polynomial size and logarithmic depth. It follows that if the reduction is computable in logspace-uniform $\mathrm{NC}^1$, then P equals logspace-uniform $\mathrm{NC}^1$.

OVERVIEW. We begin with the observation that the bottleneck in the randomized and deterministic $NC^2$ algorithms of the previous sections is the solution of system of linear equations over $GF(2)$. This is a purely technical matter, and does not seem to be in any way relevant to the validity of the conjecture. Thus we are left with the problem of identifying some more structure in the equations we produce, and find a way to take advantage of it. Again, it turns out that the problem itself points to the solution. If we spend a moment investigating the structure of the set of equations produced using the deterministic construction, we note that it is very structured, and is highly suggestive of Vandermonde matrices. This naturally suggests the possibility that we can try to directly produce a Vandermonde system of equations, and hope that it will be somehow easier to solve, say in $NC^1$. It turns out that this is precisely what we do, but we are faced with numerous technical obstacles, which we solve in the course of the proof.

**Proof.** It is known that the polynomial $X^{2\cdot 3^\ell}+X^{3^\ell}+1 \in \mathbb{Z}_2[X]$ is an irreducible polynomial over $\mathbb{Z}_2$ for all $\ell \geq 0$ [vL91]. In the following, by a finite field $GF(2^m)$, where $m = 2\cdot 3^\ell$, we refer explicitly to the field $\mathbb{Z}_2[X]/(X^{2\cdot 3^\ell}+X^{3^\ell}+1)$.

Let $S$ be a sparse set hard for P under many-one reductions. With a view to creating a Vandermonde system of equations, we will consider the following refinement of the circuit-value problem. Define

$$L = \left\{ \langle C, x, 1^m, u, v\rangle \mid m = 2\cdot 3^\ell, u, v \in GF(2^m), \sum_{i=0}^{n-1} u^i g_i = v \right\},$$

where $C$ is a boolean circuit and $x$ is an input to $C$, and where $g_0, \ldots, g_{n-1}$ are 0-1 variables that denote the values of the gates of $C$ on input $x$. Here exponentiation and summation are carried out in the finite field $GF(2^m)$. It is easy to see that $L \in P$, since all the required field arithmetic involved in checking $\sum u^i g_i = v$ can be performed in polynomial time.

Clearly $|\langle C, x, 1^m, u, v\rangle|$ is bounded polynomially in $n$ and $m$. If $f$ is a logspace-computable function that reduces $L$ to $S$, the bound on the length of queries made by $f$ on inputs of length $|\langle C, x, 1^m, u, v\rangle|$ is some polynomial $q(n,m)$. Let $p(n,m)$ be a polynomial that bounds the number of strings in $S$ of length at most $q(n,m)$. We will choose the smallest $m$ of the form $2\cdot 3^\ell$ such that $2^m/p(n,m) \geq n$. It is clear that $m = O(\log n)$. Let $\mathbb{F}$ denote the finite extension $GF(2^m)$ of $GF(2)$.

*Remarks.* Alternatively, we can take $\mathbb{F}$ to be the finite field $\mathbb{Z}/(a)$ for some prime number $a$ that satisfies $a/p(n, \lceil \log_2 a \rceil) \geq n$. Our main theorem in this chapter is valid with either choice of $\mathbb{F}$. The important point is that it should be possible to implement certain basic operations, to be listed shortly, in $NC^1$. We prefer to retain $GF(2^m)$ not only because it is a natural outgrowth of the ideas from the previous section and because it simplifies exposition of Boolean complexity of the operations, but also because of the reason that the result of the Section 2.6, which builds upon ideas from this section, seems to depend critically on the choice of the field to be $\mathbb{Z}_2[X]/(X^{2\cdot 3^\ell}+X^{3^\ell}+1)$.

Our parallel algorithm for *CVP* begins by computing $f(\langle C, x, u, v\rangle)$ for all $u, v \in \mathbb{F}$. For every $u \in \mathbb{F}$, there is a unique element $v_u \in \mathbb{F}$ such that $\langle C, x, u, v_u\rangle \in L$, and therefore $f(\langle C, x, u, v_u\rangle) \in S$. Since $2^m/p(n,m) \geq n$, there is at least one string $w \in S$ such that the number of $u$ satisfying $f(\langle C, x, u, v_u\rangle) = w$ is at least $n$. Of course, there could be many such $w$ (not necessarily in $S$), and we don't know which $w$ is a string in $S$. To handle this, we will assume that every $w$ that has $\geq n$ preimages is a string in $S$, and attempt to solve for the $g_i$'s. As long as there is at least one $w \in S$ that has $\geq n$ preimages, one of the assumptions must be correct, and we will have the correct solution. Since we know the details of the circuit $C$, the solutions can be checked, and the incorrect ones weeded out.

Assume, therefore, without loss of generality, that $w \in S$ has $\geq n$ preimages. Let $u_1, u_2, \ldots, u_n$ denote $n$ of them, and let $v_1, v_2, \ldots, v_n$ denote the corresponding $v_u$'s. The equations

$$1g_0 + u_j g_1 + u_j^2 g_2 + \ldots + u_j^{n-1} g_{n-1} = v_j,$$

for $j = 1, 2, \ldots, n$, form an inhomogeneous system of linear equations, where the coefficients $(u_j^i)$ form a Vandermonde matrix, which we will denote by $U$. Since the $u_j$'s are distinct elements of the field $\mathbb{F}$, the system $Ug = v$ has full rank. It remains to show how to solve this system of equations in logspace-uniform NC$^1$. Thus the proof of Theorem 2.10 is complete, modulo the following lemma, which is of general interest. $\square$

**Lemma 2.11** *Let $\mathbb{F} = GF(2^m)$, where $m = O(\log n)$, and $m$ is of the form $2 \cdot 3^\ell$ for some integer $\ell \geq 0$. Solving a system $Ug = v$ of $n$ equations in $n$ unknowns over the field $\mathbb{F}$, where $U$ is a Vandermonde matrix of full rank over $\mathbb{F}$, can be done by an $O(\log n)$-space uniform circuit of size $n^{O(1)}$ and depth $O(\log n)$.*

**Computation in $\mathbb{F}$:** Before proceeding to the proof of Lemma 2.11, we first collect some facts about implementing the basic operations of $\mathbb{F}$. The complexity of these operations is important in determining the size, depth and the uniformity of the circuits that we build.

(1) Adding two elements $\alpha, \beta \in \mathbb{F}$ is just the bitwise exclusive-or of the representations of $\alpha$ and $\beta$, and can be done in depth $O(1)$. Adding $n^{O(1)}$-many elements can be done by a circuit of size $n^{O(1)}$ and depth $O(\log n)$, using the obvious recursive doubling strategy. The circuitry to perform these additions are clearly logspace-uniform.

(2) Multiplying two elements $\alpha, \beta \in \mathbb{F}$ can be done using $O(\log m) = O(\log\log n)$ space, or by a circuit of depth $O(\log m) = O(\log\log n)$ and size $m^{O(1)} = (\log n)^{O(1)}$, as follows. For $\gamma \in \mathbb{F}$, let $P_\gamma \in \mathbb{Z}_2[X]$ denote the polynomial whose coefficients are given by the bits of $\gamma$. Clearly, $\alpha \cdot \beta = (P_\alpha \cdot P_\beta) \bmod (X^m + X^{m/2} + 1)$. Each of the $2m - 1$ coefficients of $P_\alpha \cdot P_\beta$ is the sum (in $\mathbb{Z}_2$) of at most $m$ bits, and can be evaluated in $O(\log m)$ space, or by a circuit of size $O(m^2)$ and depth $O(\log m)$. Finally, implementing the "mod$(X^m + X^{m/2} + 1)$" operation on $P_\alpha * P_\beta$ can be done easily in $O(\log m)$ space, or by a circuit of size $O(m)$ and depth $O(\log m)$.

(3) Finding a primitive element $\omega$ that generates the multiplicative group $\mathbb{F}^*$ of $\mathbb{F}$ can be done in logspace by exhaustive search. An element $\omega \in \mathbb{F}$ generates $\mathbb{F}^*$ iff the condition "$(\forall \alpha \in \mathbb{F})(\exists i < 2^m)[\omega^i = \alpha]$" holds. The latter condition can be tested using $O(m) = O(\log n)$ space by maintaining two counters, one that runs through all elements $\alpha$ of $\mathbb{F}$, and another for the exponent $i$, and doing the multiplications as described in Fact (2). Note that finding a primitive element is part of the precomputation, and does not have to be implemented in NC$^1$.

(4) Raising the generator $\omega$ to any power $i < 2^m$, or computing the discrete logarithm of any element with respect to $\omega$, can be done by table lookup in depth $O(\log n)$. The tables themselves can be precomputed using $O(\log n)$ space.

(5) The following fact is less obvious, and will be important: multiplying $k = n^{O(1)}$ elements of $\mathbb{F}$ can be done in $O(\log n)$ depth. Given elements $\alpha_1, \alpha_2, \ldots, \alpha_k$, first the discrete logarithms $\ell_1, \ell_2, \ldots, \ell_k$ of the $k$ elements are computed with respect to the generator $\omega$. By Fact (4), this can be done simultaneously in $O(\log n)$ depth and size $n^{O(1)}$. The next task is to add the $k$ $O(\log n)$-bit integers $\ell_1, \ell_2, \ldots, \ell_k$, and reduce the sum modulo $2^m - 1$. The addition can be done in $O(\log n)$ depth using the folklore 3-to-2 trick, in the following manner. Divide the $k$ integers into $\lceil k/3 \rceil$ groups of three integers each. By computing the "sum" and the "carry" parts of the addition separately, the three integers $\ell_i, \ell_{i+1}, \ell_{i+2}$ in the $i$-th group can be converted into two integers $\ell'_i$ and $\ell''_i$, such that $\ell_i + \ell_{i+1} + \ell_{i+2} = \ell'_i + \ell''_i$. Moreover, this can be accomplished in depth $O(1)$ simultaneously for all groups of three elements, thus producing a list of $2k/3$ elements whose sum equals the sum of the $k$ elements $\ell_1, \ell_2, \ldots, \ell_k$. By recursively applying this idea, the sum of the $k$ integers can be computed in depth $O(\log_{3/2} k) = O(\log n)$. Since the sum of the $k$ integers is at most $k2^m = n^{O(1)}$, reducing the sum modulo $2^m - 1$ can be easily accomplished by a table look-up in depth $O(\log n)$. It is also clear that the look-up table can be precomputed in space $O(\log n)$. Finally, converting the discrete logarithm into the corresponding field element can be done by table look-up in depth $O(\log n)$.

**Proof.** (of Lemma 2.11) Observe that an equation of the form $\sum_{j=0}^{n-1} g_j u^j = v$ can be viewed as specifying the value of the polynomial $G(u) \doteq \sum_{j=0}^{n-1} g_j u^j$ at the point $u \in \mathbb{F}$. With this viewpoint, our task is to infer the polynomial $G$, that is, to find the coefficients $g_j$ of $G$. Clearly if we can evaluate $G(u)$ at $n$ distinct points $u_1, \ldots, u_n \in \mathbb{F}$, then we can recover the coefficients $g_j$ by Lagrange interpolation as follows:

$$G(u) = \sum_{i=1}^{n} G(u_i) Q_i = \sum_{i=1}^{n} v_i Q_i$$

where

$$Q_i = \frac{(u-u_1)\ldots(u-u_{i-1})(u-u_{i+1})\ldots(u-u_n)}{(u_i-u_1)\ldots(u_i-u_{i-1})(u_i-u_{i+1})\ldots(u_i-u_n)} = \prod_{k \neq i} \frac{(u-u_k)}{(u_i-u_k)}.$$

For $0 \leq j < n$, $g_j$ is the coefficient of $u^j$ in $G(u)$. Collecting the terms corresponding to $u^j$, we have

$$g_j = \sum_{i=1}^{n} (-1)^{i+1} \frac{v_i}{\prod_{k \neq i}(u_k - u_i)} P_{n-j-1}(u_1, \ldots, \widehat{u_i}, \ldots, u_n).$$

Here $\widehat{u_i}$ denotes that $u_i$ is missing from the list $u_1, \ldots, u_n$, and $P_k$ denotes the $k$-th elementary symmetric polynomial, defined as follows:

$$P_0(y_1, \ldots, y_\ell) = 1; \qquad P_k(y_1, \ldots, y_\ell) = \sum_{\substack{I \subseteq [\ell] \\ |I| = k}} \prod_{i \in I} y_i, \qquad k > 0.$$

By Facts (2) and (5), computing $v_i / (\prod_{k \neq i}(u_k - u_i))$ in NC$^1$ is fairly straightforward. Hence it suffices to show how to compute the polynomials $P_k(u_1, \ldots, \widehat{u_i}, \ldots, u_n)$, in logspace-uniform NC$^1$. A folklore theorem indicates that this can be done in non-uniform NC$^1$. For our application, however, the uniformity is crucial.

It is easy to see that for $y_1, \ldots, y_\ell \in \mathbb{F}$, $P_k(y_1, \ldots, y_\ell)$ equals $P_k(y_1, y_2, \ldots, y_\ell, 0, 0, \ldots, 0)$ for any number of extra zeroes. Let $r = |\mathbb{F}^*|$, the number of elements in the multiplicative group of $\mathbb{F}$. We will give an $\mathrm{NC}^1$ algorithm to compute the elementary symmetric polynomial of $r$ elements, not necessarily distinct, from the finite field $\mathbb{F}$. By appending $r-\ell$ zeroes, we can then compute $P_k(y_1,y_2,\ldots,y_\ell)$.

For $0 < k \le r$, the value of the elementary symmetric polynomial $P_k(y_1,y_2,\ldots,y_r)$ is the coefficient of $X^{r-k}$ in $h(X) \doteq \prod_{i=1}^{r}(X+y_i) - X^r$. Note that, given any $\alpha \in \mathbb{F}$, $h(\alpha)$ can be evaluated in $\mathrm{NC}^1$, by Facts (1) and (5).

If we write $h(X)$ as $\sum_{i=0}^{r-1} a_i X^i$, the coefficient $a_i = P_{r-i}(y_1,\ldots,y_r)$ for $0 \le i < r$. The idea now is to choose $\alpha$'s carefully from $\mathbb{F}$, compute $h(\alpha)$ and compute the coefficients $a_i$ by interpolation. If we choose $\omega$ to be a primitive element of order $r$ in $\mathbb{F}^*$, the powers of $\omega$, namely $1=\omega^0,\omega^1,\omega^2,\ldots,\omega^{r-1}$, run through the elements of $\mathbb{F}^*$. For $0 \le i < r$, let $b_i = h(\omega^i)$. The relationship between the pointwise values ($b_i$'s) and the coefficients ($a_i$'s) of $h(X)$ can be written as:

$$\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{r-1} \end{pmatrix} = \begin{pmatrix} 1 & \omega^0 & \omega^{0\cdot 2} & \ldots & \omega^{0\cdot(r-1)} \\ 1 & \omega^1 & \omega^{1\cdot 2} & \ldots & \omega^{1\cdot(r-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{r-1} & \omega^{(r-1)\cdot 2} & \ldots & \omega^{(r-1)\cdot(r-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{r-1} \end{pmatrix}.$$

The above matrix, which we will denote by $\Omega$, is the Discrete Fourier Transform matrix, and is a Vandermonde matrix. Since the powers of $\omega$ are all distinct, $\Omega$ is invertible, and one can compute the coefficients $a_i$ by $(a_0,\ldots,a_{r-1})^T = \Omega^{-1}(b_0,\ldots,b_{r-1})^T$. The crucial advantage over the earlier Vandermonde system is that with this particular choice of $\Omega$, the matrix $\Omega^{-1}$ has a simple explicit form: the $(i,j)$-th entry of $\Omega^{-1}$ is just $\omega^{-ij}$. Computing the coefficients of $h(X)$ is now simply a matrix-vector multiplication. This completes the proof of the lemma. $\square$

**Corollary 2.12 (Theorem 2.3)** *If there is a sparse set S that is hard for* P *under logspace-computable many-one reductions, then* P = LOGSPACE.

**Corollary 2.13** *If there is a sparse set S that is hard for* P *under many-one reductions computable in logspace-uniform* $\mathrm{NC}^1$*, then* P *equals logspace-uniform* $\mathrm{NC}^1$.

**Corollary 2.14** *If there is a set S such that for all n,* $|S\cap\Sigma^n| \le 2^{(\log n)^a}$ *that is hard for* P *under many-one reductions computable in space* $(\log n)^b$*, then* $\mathrm{P} \subseteq \mathrm{DSPACE}[(\log n)^{ab}]$.

## 2.5 Sparse Hard Sets under Randomized Reductions

In this section, we consider many-one-style reductions to sparse sets that are *probabilistic*, that is, the reduction is allowed make some mistakes. We begin with the following definition.

**Definition 2.6 (BP-Reduction)** *A language A is said to be* ***randomly many-one reducible to** B **with two-sided error*** *(or* ***bp-reducible****) if there is a function* $f(\cdot,\cdot)$ *and some fixed* $\varepsilon > 0$ *such that for all x,*

$$x \in A \iff \Pr_z[f(x,z) \in B] \ge \frac{1}{2} + \varepsilon.$$

We consider the hypothesis that there is a sparse set that is hard for P under randomized reductions (with two-sided error) that are computable in logspace or in $\mathrm{NC}^1$. Our main result in this section is the following:

**Theorem 2.15** *If there is a sparse set $S$ that is hard for* P *under randomized many-one reductions with two-sided error, then the* P*-complete circuit-value problem (CVP) can be solved by a logspace-uniform family of randomized circuits of polynomial size and logarithmic depth that make polynomially many parallel calls to the reduction.*

**Corollary 2.16** *If there is a sparse set $S$ that is hard for* P *under randomized reductions computable in logspace-uniform* $\mathrm{NC}^1$*, then* $\mathrm{P} \subseteq \mathrm{RNC}^1$.

**Corollary 2.17** *If there is a sparse set $S$ that is hard for* P *under logspace computable randomized reductions with two-sided error, then* $\mathrm{P} = \mathrm{RL}_2$*, where* $\mathrm{RL}_2$ *is the class of languages accepted by randomized logspace Turing machines that have two-way access to their random tapes.*

Corollary 2.17 follows from the fact that the simulation of an $O(\log n)$-depth randomized circuit can be done by a logspace randomized Turing machine that has *two-way access* to its random tape. The class $\mathrm{RL}_2$ is considered to be more powerful than the standard definition of randomized logspace (RL), where the logspace-bounded Turing machine is only allowed one-way access to the random tape. See [Nis93] for more on the distinction between the two models.

OVERVIEW OF THE PROOF. At first glance, one's initial reaction to this theorem would be that it may be proved following the proof of Theorem 2.10, taking into account the fact that the reduction is randomized. After all, the result has the same flavor: Theorem 2.10 says that if the many-one reduction to the sparse set is deterministic, then P collapses to $\mathrm{NC}^1$ and Theorem 2.5 says that if the many-one reduction to the sparse set is randomized, then P collapses to $\mathrm{RNC}^1$. However, it turns out not to be so simple. It is possible to prove—and we indeed do so—that there is a "popular" $w^* \in S$ that is also fairly "reliable" in that at least $1/2 + \delta$ fraction of the equations we can produce with respect to $w^*$ are correct for some $\delta = \delta(\varepsilon) > 0$. This is where we run into a problem: it is not clear how to solve a *given* set of linear equations over a finite field when only a $(1/2 + \delta)$-fraction of the equations are guaranteed to be correct, and what is worse, this is believed to be cryptographically hard when the field is just $\mathbb{Z}_2$ and the errors are chosen at random [BFKL94] with some constant probability.

Once again, the problem hints at the solution. Above, we emphasized the fact that we have no control over the equations produced. What if we do have the luxury of asking questions of our choice about a hidden function (the function $g = g(C,x)$)? Can we do better? Let us try the simple case where $g$ is a linear function on $n$ variables over $\mathbb{Z}_2$, as in Section 2.2. We ask the following question:

> Let $g(Z_1, Z_2, \ldots, Z_n) = (\sum_i g_i Z_i) \bmod 2$ be an $n$-variate linear function over $\mathbb{Z}_2$. Suppose $G(\cdot,\cdot)$ is a probabilistic oracle such that $\Pr_{Z \in \mathbb{Z}_2^n, z}[G(Z,z) = g(Z)]$ is at least $1/2 + \delta$ for some constant $\delta > 0$. Can we recover the linear function $g$ (i.e. the bits $g_i$) by making at most a polynomial number of queries to $G$?

It turns out that this problem has an elegant solution due to Goldreich and Levin [GL89] (see also [Gol95]). They considered the case that $G$ is a deterministic oracle and the probability was only over the $Z$'s, but their algorithm, in fact, works even if $G$ is probabilistic. Moreover, their algorithm can be implemented in $\mathrm{NC}^1$!

Now our task is simplified, but not completed: we still have to come up with a way of creating the oracle $G$. This is where we need the notion of a popular $w^* \in S$ that is also reliable. The proof that follows makes this notion precise and constructs the oracle $G$.

**Proof.** (of Theorem 2.15) As in the earlier proofs, we will define a language $A \in \mathrm{P}$, and using the reduction $f$ from $A$ to the sparse set $S$ show how to solve $CVP$ in $\mathrm{RNC}^1$. The language $A$ consists of tuples $\langle C,x,Z,b\rangle$, where $C$ is a boolean circuit with $n$ gates (including the input and output gates), $x$ is an input to $C$, $Z \in \mathbb{Z}_2^n$ denotes a subset of $\{g_1,\ldots,g_n\}$, and $b \in \mathbb{Z}_2$, such that $g(Z) = b$. Here $g(Z)$ denotes the linear function over $\mathbb{Z}_2$ given by $(Z_1,Z_2,\ldots,Z_n) = (\sum g_i Z_i) \bmod 2$ where, for $1 \le i \le n$, $g_i$ denotes the value of the $i$-th gate of $C$ when the input is $x$.

Fix $C$ and $x$. As in the earlier proofs, let $p = p(n)$ denote the polynomial bound on the number of strings in $S$ than can be produced as images of the function $f$ that reduces $A$ to the sparse set $S$. Suppose the reduction $f$ uses $r$ random bits on inputs of the form $\langle C,x,Z,b\rangle$ and achieves a success probability of $\frac{1}{2}+\varepsilon$; that is,

$$\langle C,x,Z,b\rangle \in L \iff \Pr_{z\in\{0,1\}^r}[f(\langle C,x,Z,b\rangle,z) \in S] \ge \frac{1}{2}+\varepsilon,$$

for some constant $\varepsilon$. In fact, it is not hard to see that the proof works even if $\varepsilon$ is only $1/n^{O(1)}$.

The values $g_1,\ldots,g_n$ define a linear function $g : \mathbb{Z}_2^n \to \mathbb{Z}_2$, and our objective is to reconstruct this function, that is, find the coefficients $g_i$. Recall that our strategy to accomplish this is to build a probabilistic oracle $G : \mathbb{Z}_2^n \to \mathbb{Z}_2$ that achieves some non-trivial advantage (over random guessing) in predicting the value of the function $g$.

(1) We will first show that there is a string $w^* \in S$ such that:

(a) $w^*$ is *popular*, that is,

$$\Pr_{Z,b,z}[f(\langle C,x,Z,b\rangle,z) = w^*] > \frac{\varepsilon}{2p} = \frac{1}{n^{O(1)}}.$$

(b) $w^*$ is *reliable*, that is,

$$\Pr_{Z,b,z}[\langle C,x,Z,b\rangle \in A \mid f(\langle C,x,Z,b\rangle,z) = w^*] > \frac{1}{2}+\frac{\varepsilon}{2}.$$

(2) Using $w^*$, we then build the probabilistic oracle $G$ that attempts to compute the linear function $g$ in the following manner. $G$ takes $Z \in \{0,1\}^n$ as input, flips coins for $b$ and $z$, computes $w = f(\langle C,x,Z,b\rangle,z)$. If $w = w^*$, then $G$ outputs $b$ as the value of $g(Z)$; else, $G$ outputs the outcome of a coin flip (0 or 1 with equal probability) as the value of $g(Z)$.

Of course, we don't know what $w^*$ is. To circumvent this, we make polynomially many trials with randomly chosen $Z$'s to identify all popular $w$'s—so long as $w^*$ is popular, it will appear in our trials with very high probability. For each popular $w$, we will assume that $w^* = w$ and proceed with our computations (in parallel). We will argue that for the correct candidate for $w^*$, our algorithm will reconstruct $g$ whp., while the wrong output from every incorrect candidate for $w^*$ can be identified and eliminated using local information about the circuit $C$. Hence we will describe the remainder of the algorithm assuming that the correct value of $w^*$ is available.

We now show that the oracle $G$ achieves non-negligible advantage (over random guessing) in computing the linear function $g$. More precisely, we have

$$\begin{aligned}
& \Pr_{Z,b,z}[G(Z,b,z) = g(Z)] \\
= \; & \Pr_{Z,b,z}[f(\langle C,x,Z,b\rangle,z) = w^*] \cdot \Pr_{Z,b,z}[G(Z,b,z) = g(Z) \mid f(\langle C,x,Z,b\rangle,z) = w^*] \\
+ \; & \Pr_{Z,b,z}[f(\langle C,x,Z,b\rangle,z) \neq w^*] \cdot \Pr_{Z,b,z}[G(Z,b,z) = g(Z) \mid f(\langle C,x,Z,b\rangle,z) \neq w^*] \\
> \; & \Pr_{Z,b,z}[f(\langle C,x,Z,b\rangle,z) = w^*] \cdot \left(\frac{1}{2} + \frac{\varepsilon}{2}\right) + \left(1 - \Pr_{Z,b,z}[f(\langle C,x,Z,b\rangle,z) = w^*]\right) \cdot \frac{1}{2} \\
= \; & \frac{1}{2} + \frac{\varepsilon}{2} \cdot \Pr_{Z,b,z}[f(\langle C,x,Z,b\rangle,z) = w^*] \\
> \; & \frac{1}{2} + \frac{\varepsilon^2}{4p}.
\end{aligned}$$

Letting $\delta = \delta(\varepsilon) = \varepsilon^2/(4p) = 1/n^{O(1)}$, $G$ has a probabilistic advantage of $\delta$ in predicting $g$, where the probability is over all inputs $Z$ and the internal coin tosses of $G$. This completes the proof, modulo the existence of a string $w^*$ that is both popular and reliable. We proceed to establish this now.

For $Z \in \mathbb{Z}_2^n$, let $b_Z = g(Z) \doteq g(Z)$, with notations as before. Let $r$ denote the number of random bits needed for the reduction $f$. We define matrices $P$ and $Q$, both of size $2^n \times 2^r$, indexed by strings of the form $(Z,z)$, where $Z \in \mathbb{Z}_2^n$ and $z \in \{0,1\}^r$, and where:

$$P[Z,z] = f(\langle C,x,Z,b_Z\rangle,z); \qquad Q[Z,z] = f(\langle C,x,Z,\widetilde{b_Z}\rangle,z).$$

In other words, $P(Z,z)$ gives the string produced by the reduction $f$ on input $Z$ with random string $z$, for the "correct" answer $b_Z$, and $Q(Z,z)$ gives the string produced for the "wrong" answer $\widetilde{b_Z}$.

For $1 \leq i \leq p = p(n)$, let $w_i$ denote the $i$-th string in the sparse set $S$ (of the appropriate length), let $P_i$ denote the number of times $w_i$ appears in the matrix $P$, and let $Q_i$ denote the number of times $w_i$ appears in the matrix $Q$. By the property of the reduction, we know

$$\left(\frac{1}{2} + \varepsilon\right) 2^{n+r} \leq \sum_i P_i \leq 2^{m+r}; \qquad 0 \leq \sum_i Q_i \leq \left(\frac{1}{2} - \varepsilon\right) 2^{n+r}.$$

A moment's reflection indicates that the quantity $\sum_i P_i / (\sum_i P_i + \sum_i Q_i)$ is at least $\frac{1}{2} + \varepsilon$: Clearly, it is minimized when $\sum_i Q_i$ is at its maximum, which is $(\frac{1}{2} - \varepsilon)2^{m+r}$. Moreover, the function $\xi/(\xi + \alpha)$ is strictly increasing in the interval $(\frac{1}{2} + \varepsilon) \leq \xi \leq 1$, for constant $\alpha > 0$, and hence attains its minimum when $\xi$ is at its minimum, namely at $(\frac{1}{2} + \varepsilon)$.

Also, for $1 \le i \le p$, let $N_i = P_i + Q_i$, the *total* number of times $w_i$ appears in the matrices $P$ and $Q$, and let $R_i = P_i/(P_i + Q_i)$, the ratio of the number of "correct" appearances of $w_i$ to its total number of appearances. From the preceding calculations, we know

$$\frac{\sum_i N_i R_i}{\sum_i N_i} = \frac{\sum_i P_i}{\sum_i P_i + \sum_i Q_i} \ge \frac{1}{2} + \varepsilon.$$

This is precisely the expectation of $R_i$ when a random element $w_i \in S$ is chosen uniformly from all the occurrences of strings in $S$ in the matrices $P$ and $Q$ together. Note that a string $w_i$ is *reliable* if $R_i > \frac{1}{2} + \frac{\varepsilon}{2}$.

Let $k$ denote the number of reliable strings, and wlog., let $w_1, \ldots, w_k$ denote the reliable strings. From the fact that $E[R_i] \ge \frac{1}{2} + \varepsilon$, it is immediate that $k \ge 1$; what we want in addition is that for some $i \le k$, $N_i$ is large so that $w_i$ is also *popular*. Now

$$N_1R_1 + N_2R_2 + \ldots + N_kR_k + N_{k+1}R_{k+1} + \ldots + N_pR_p = \sum_{i=1}^{p} N_iR_i \ge \left(\frac{1}{2} + \varepsilon\right) \sum_i N_i.$$

Consider an adversary who attempts to prevent the existence of a popular string $w_i, i \le k$. In order to satisfy the lower bound of $(\frac{1}{2} + \varepsilon) \sum_i N_i$, his best strategy would be to set the $R_i$'s, for $1 \le i \le k$, to their maximum value 1. For $i > k$, the $w_i$'s are not reliable, hence the maximum he can set these $R_i$'s to is $\frac{1}{2} + \frac{\varepsilon}{2}$. This shows that

$$(N_1 + \ldots + N_k)\left(\frac{1}{2} + \frac{\varepsilon}{2} + \frac{1}{2} - \frac{\varepsilon}{2}\right) + (N_{k+1} + \ldots + N_p)\left(\frac{1}{2} + \frac{\varepsilon}{2}\right) \ge \left(\frac{1}{2} + \varepsilon\right) \sum_i N_i,$$

and hence we have

$$\left(\frac{1}{2} - \frac{\varepsilon}{2}\right)(N_1 + \ldots + N_k) \ge \frac{\varepsilon}{2} \sum_i N_i.$$

It follows from routine calculations and by the pigeonhole principle that for some $i^* \le k$, $N_{i^*} > \varepsilon 2^{m+r}/(2p)$. Also, since $w^* = w_{i^*}$ is reliable ($i^* \le k$), we know that $R_{i^*} > \frac{1}{2} + \frac{\varepsilon}{2}$. □

### 2.5.1 Application to NP

Theorem 2.15 has an interesting application for the class NP. Recall that Mahaney [Mah82] showed that NP cannot have a sparse hard set under polynomial-time many-one reductions, and that nearly a decade later, Ogihara and Watanabe [OW91] introduced a new technique to extend Mahaney's result to the case of the so-called bounded truth-table reductions. Extending their technique, Ranjan and Rohatgi [RR92] showed that NP cannot have a sparse hard set under polynomial-time computable co-rp reductions (defined below) unless NP = RP.

**Definition 2.7 (co-RP Reduction)** *A language A is said to be* ***co-rp reducible*** *to B if there is a function $f(\cdot,\cdot)$ and some fixed $\varepsilon > 0$ such that for all x,*

$$x \in A \Rightarrow \Pr_z[f(x,z) \in B] = 1; \qquad x \notin A \Rightarrow \Pr_z[f(x,z) \notin B] \ge \frac{1}{2} + \varepsilon.$$

The complementary kind of reductions are called *rp-reductions*, and are defined as follows:

**Definition 2.8 (RP Reduction)** *A language $A$ is said to be* ***rp reducible*** *to $B$ if there is a function $f(\cdot,\cdot)$ and some fixed $\varepsilon > 0$ such that for all $x$,*

$$x \in A \Rightarrow \Pr_z[f(x,z) \in B] \geq \frac{1}{2} + \varepsilon. \qquad x \notin A \Rightarrow \Pr_z[f(x,z) \notin B] = 1;$$

The question of whether the existence of a sparse hard set for NP under rp-reductions (or under bp-reductions) implies NP = RP has remained open. It is clear that bp-reductions include both rp and co-rp reductions. Using the techniques of Theorem 2.15 together with a lemma of Valiant and Vazirani [VV86], we resolve this question in the affirmative.

**Theorem 2.18** *If there is a sparse set $S$ that is hard for* NP *under polynomial time computable bp-reductions with $\varepsilon = 1/n^{O(1)}$, then* NP = RP.

**Proof Sketch.** Under the hypothesis that NP is bp-reducible to a sparse set, we show that $SAT \in$ RP. Given a boolean formula $\varphi(x_1, x_2, \ldots, x_n)$, we first apply the reduction of Valiant and Vazirani [VV86] to produce a list of boolean formulae $\varphi_1, \varphi_2, \ldots, \varphi_k$, where $k = n^{O(1)}$, and where each $\varphi_i$ is a formula in $n^{O(1)}$ variables. This reduction has the property that if $\varphi$ is satisfiable, then with probability $1 - e^{-n}$, at least one of the $\varphi_i$'s has a *unique* satisfying assignment, and if $\varphi$ is not satisfiable, then no $\varphi_i$ is satisfiable. Assume that $\varphi^*$ is a boolean formula in $t$ variables that has a unique satisfying assignment; it suffices to show how to construct this assignment using the reduction of NP to a sparse set $S$. We define the language

$$L = \left\{ \langle \psi, Z, b \rangle \mid Z \in \mathbb{Z}_2^n, b \in \mathbb{Z}_2, (\exists \vec{a} = (a_1, \ldots, a_t))[\psi(\vec{a}) \wedge \sum_{i=0}^{t-1} a_{i+1} Z_i = b] \right\}.$$

Clearly $L \in$ NP, therefore $L$ reduces to $S$. Now, using the same ideas as in the case of bp-reductions from P to a sparse set, we can construct the satisfying assignment $\vec{a}^*$ of $\varphi^*$ in randomized polynomial time. Of course, any unsatisfying assignment $\vec{a}$ produced in the process can be easily weeded out by checking if $\varphi(\vec{a}) =$ TRUE. $\square$

## 2.6 Sparse Hard Sets for NL

In this section, we show the resolution of the sparse set conjecture of Hartmanis for the complexity class NL. We show that there is a sparse hard set for NL under logspace many-one reductions iff NL = L. Our proof uses the algebraic techniques of Section 2.4. An additional crucial ingredient in the proof is the famous result of Immerman [Imm88] and Szelepcsényi [Sze87], that NL = co-NL.

**Theorem 2.19** *If there is a sparse set $S$ that is hard for* NL *under logspace many-one reductions, then there is an* NL*-complete problem that can be solved by a logspace-uniform family of polynomial size, logarithmic depth circuits that make polynomially many parallel queries to the reduction from* NL *to $S$.*

That is, modulo the complexity of the reduction, the parallel algorithm works in $\mathrm{NC}^1$. It follows that if there is a sparse hard set for NL under many-one reductions computable in logspace-uniform $\mathrm{NC}^1$, then NL equals logspace-uniform $\mathrm{NC}^1$, and that if there is a sparse hard set for NL under many-one reductions computable in logspace, then $\mathrm{NL} = \mathrm{L}$.

### 2.6.1 A Complete Problem for NL

Given a directed graph $G = (V, E)$ and two distinguished vertices $s, t \in V$, the *s-t connectivity problem* asks whether there is a directed path from $s$ to $t$ in $G$, i.e. whether a sequence of directed edges $(s, u_1), (u_1, u_2), \ldots, (u_k, t)$ exists. The *s-t* connectivity problem is well-known to be complete for NL under logspace many-one reductions [Sav73]. Immerman [Imm87] has shown that this problem is complete for NL under an extremely weak form of many-one reductions called quantifier-free first-order projections. We note that the *s-t* connectivity problem is complete for NL (under logspace- or even $\mathrm{NC}^1$-computable) many-one reductions, even when restricted to directed acyclic graphs. We call this problem "*DAG-STCON*." Moreover, without loss of generality, we may also assume that all instances of *DAG-STCON* are labeled and layered graphs, that is, graphs where all edges go from lower-numbered vertices to higher-numbered vertices. A main consequence of the completeness of *DAG-STCON* is that if *DAG-STCON* $\in$ logspace-uniform $\mathrm{NC}^1$, then $\mathrm{NL} =$ logspace-uniform $\mathrm{NC}^1$, and if *DAG-STCON* $\in \mathrm{L}$, then $\mathrm{NL} = \mathrm{L}$.

### 2.6.2 The Strategy and the Proof

Let us quickly recall the main idea behind the proof of Theorem 2.10. We considered an arbitrary instance $\langle C, x\rangle$ of the P-complete circuit-value problem, and devised a way of asking exponentially many questions about the solution to this instance. To facilitate this, we viewed the "certificate" of the computation of $C$ on $x$ as an $n$-bit string that specifies the coefficients of a polynomial. When we consider the case of NL, for an instance $\langle G, s, t\rangle$ of *DAG-STCON*, there could be exponentially many certificates that demonstrate that $t$ is reachable from $s$. It is not clear which of these certificates we should use to define the coefficients of the polynomial. If we decide to use, for example, the lexicographically first certificate, then the resulting auxiliary language (such as "$A$" in the proof of Theorem 2.10) seems no longer to be in NL.

We did encounter this problem when we applied the strategy of the proof of Theorem 2.15 to the case of NP, which also has the problem of potential multiple certificates. There we appealed to the randomized reduction of Valiant and Vazirani [VV86] to produce an instance of *SAT* that has a unique assignment. We can follow a similar approach here, using the randomized reduction [Wig94] from $s$-$t$-connectivity to "Unique $s$-$t$-connectivity." That would imply that if NL has a sparse hard set under logspace many-one reductions, then $\mathrm{NL} \subseteq \mathrm{RL}_2$, where $\mathrm{RL}_2$ is the class of languages accepted by logspace bounded randomized Turing machines that have two-way access to their random tapes. This, of course, is inadequate to resolve the conjecture about sparse hard sets for NL.

However, there is a wonderful advantage with NL that we do not have with NP. By the famous result of Immerman [Imm88] and Szelepcsényi [Sze87], we know that $\mathrm{NL} = \text{co-NL}$. We exploit this fact to define a "certificate" for an NL computation that is unique for each instance of the *DAG-STCON* problem. With this new notion of a certificate of an NL computation, checking the validity of a certificate becomes slightly more complicated. Moreover, even after the certificate has been defined, there are many technicalities (such as evaluating a polynomial over a finite field) that become complicated because of the limitation on the space bound. The proof shows how to address these issues carefully.

**Proof.** (of Theorem 2.19) Following the ideas of Section 2.4, we will define an auxiliary language in NL. Write $k = \binom{n}{2}$. Let $B$ be the set of all tuples of the form $\langle G, s, t, 1^m, \alpha, \alpha^2, \ldots, \alpha^{k-1}, \beta \rangle$, where:

(1) $G = (V, E)$ is a directed, layered acyclic graph on $n$ vertices, and has at most $k = \binom{n}{2}$ edges. Hence the adjacency matrix $A_G$ of $G$ is upper-triangular.

(2) $s$ and $t$ are vertices in $G$.

(3) $m$ is of the form $2 \cdot 3^\ell$ for some integer $\ell \geq 0$.

(4) $\alpha, \beta \in GF(2^m)$.

(5) For $1 \leq i < k$, $\alpha^i$ denotes the $i$-th power of $\alpha$ in $GF(2^m)$.

(6) $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i} = \beta$, where for $0 \leq i < k$, $0 \leq u_i < v_i < n$, $(u_i, v_i)$ denotes the $i$-th edge in $G$, and $X_{u_i v_i}$ is a boolean variable that is 1 if and only if there is a path from $u_i$ to $t$ whose first step is the edge $(u_i, v_i)$.

This definition is somewhat complicated by the necessity of certain uniformity considerations. Intuitively it may appear that in place of $\langle G, s, t, 1^m, \alpha, \alpha^2, \ldots, \alpha^{k-1}, \beta \rangle$, we should have been able to use just $\langle G, s, t, 1^m, \alpha, \beta \rangle$. Unfortunately this is not so simple because with that definition, it is not clear how we can establish $B \in$ NL. However, for readability, we will abbreviate the tuple $\langle G, s, t, 1^m, \alpha, \alpha^2, \ldots, \alpha^{k-1}, \beta \rangle$ by $\langle G, s, t, 1^m, \alpha, \beta \rangle$ throughout this section.

**Claim 2.20** $B \in$ NL.

*Proof of Claim.* We will build a nondeterministic logspace machine $N$ that accepts $B$. First we argue that, in $O(\log n + \log m)$ space, $N$ may deterministically verify that the values $\alpha^2, \alpha^3, \ldots, \alpha^{k-1}$ are indeed the correct powers of $\alpha$. To do this, $N$ proceeds sequentially, for $i$ from 1 up to $k-2$, verifying the validity of $\alpha \cdot \alpha^i = \alpha^{i+1}$. For a fixed $i$, $N$ accomplishes this by sequentially computing each bit of $\alpha^{i+1}$ from the values of $\alpha$ and $\alpha^i$ given in the input, and checking it against the value of $\alpha^{i+1}$ given in the input. This requires two counters, one that can count up to $k-2$ and one that can count up to $m$. The counters can be implemented in space $O(\log n + \log m)$. Now to check the bits: For $\gamma \in GF(2^m)$, let $(\gamma)_j$ denote the $j$-th bit of $\gamma$, and let $P_\gamma \in \mathbb{Z}_2[x]$ denote the polynomial whose coefficients are given by the bits of $\gamma$. Thus, $\alpha^{i+1} = (P_\alpha \cdot P_{\alpha^i}) \bmod (x^m + x^{m/2} + 1)$, where $P_\alpha$ and $P_{\alpha^i}$ are multiplied in the ring $\mathbb{Z}_2[x]$. For $0 \leq j \leq 2m-2$, the coefficient of $x^j$ in $(P_\alpha \cdot P_{\alpha^i})$ is given by $\sum_{\sigma+\tau=j} (\alpha)_\sigma (\alpha^i)_\tau$. This is a mod 2 sum of at most $m$ bits. Denote this sum by $S(j)$. When the product $(P_\alpha \cdot P_{\alpha^i})$ is reduced modulo $x^m + x^{m/2} + 1$, the $j$-th bit of $\alpha^{i+1}$, for $0 \leq j < m$, is given by the sum, in $\mathbb{Z}_2$, of the following four contributions $S_1(j), S_2(j), S_3(j), S_4(j)$.

(1) For $0 \leq j < m$, $S_1(j) = S(j)$. This contribution comes from the term $x^j$ of the product $(P_\alpha \cdot P_{\alpha^i})$.

(2) For $0 \leq j < m/2$, $S_2(j) = S(j+m)$. This contribution comes from the term $x^\tau$, where $m \leq \tau < 3m/2$, of the product $(P_\alpha \cdot P_{\alpha^i})$.

(3) For $m/2 \le j < m$, $S_3(j) = S(j + m/2)$. This contribution also comes from the term $x^\tau$, where $m \le \tau < 3m/2$, of the product $(P_\alpha \cdot P_{\alpha^i})$.

(4) For $0 \le j < m/2$, $S_4(j) = S(j + 3m/2)$. This sum equals the coefficient of the term $x^{m+m/2+j}$ of the product $(P_\alpha \cdot P_{\alpha^i})$, which is equal, mod $x^m + x^{m/2} + 1$, to $x^j$.

Clearly, each of these sums can be evaluated in space $O(\log m)$.

Now, we may assume that the input is legitimate, that is, all the powers of $\alpha$ are correctly presented. Testing whether $\langle G, s, t, 1^m, \alpha, \beta\rangle \in B$ requires computing polynomially many predicates $X_{uv}$. The language $Z = \{\langle G, s, t, u, v\rangle \mid X_{uv} = 1\}$ is in NL, and since NL = co-NL, its complement $Z^c$ is also in NL. The nondeterministic logspace machines for $Z$ and $Z^c$ can be used to build a nondeterministic logspace machine that computes $X_{uv}$ in the following strong sense: every computation either outputs the correct value of $X_{uv}$ or aborts in a "DON'T KNOW" state, and at least one computation is guaranteed to output the correct value of $X_{uv}$.

Using this, we will build the nondeterministic $O(\log n + \log m)$ space-bounded machine $N$ that accepts $B$ as follows: Since the elements of the field $GF(2^m)$ have $m$-bit representations, the machine $N$ cannot write down entries of the field explicitly in its workspace during the computation to check $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i} = \beta$. Instead, it maintains a $(\log m)$-bit counter that checks, bit by bit, if the above equality holds. To check the equality of the $j$-th bit of $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i}$ and $\beta$, the machine $N$ proceeds as follows: $N$ first initializes a bit $b_j = 0$. Then, sequentially and nondeterministically $N$ computes $X_{u_i v_i}$ for each edge $(u_i, v_i)$. If $X_{u_i v_i} = 0$, it goes on to compute the next value $X_{u_{i+1} v_{i+1}}$. If $X_{u_i v_i} = 1$, then it finds the $j$-th bit $(\alpha^i)_j$ of $\alpha^i$ (which is present in the input), and updates $b_j = b_j \oplus (\alpha^i)_j$. Notice that, by design, every computation path of $N$ either computes $X_{u_i v_i}$ correctly (with a "certificate") and proceeds, or it aborts. Finally, $N$ accepts $\langle G, s, t, 1^m, \alpha, \beta\rangle$ if and only if for all $j$, the $j$-th bits of $\sum_{i=0}^{k-1} \alpha^i X_{u_i v_i}$ and $\beta$ match.

**End of Proof of Claim.**

By hypothesis, $B \le_m S$. Let $f$ denote the (logspace- or NC$^1$-computable) function that reduces $B$ to $S$. We will show how to solve *DAG-STCON* using $f$ as an oracle. Fix $G = (V, E)$, $s$ and $t$. Let $n = |V|$ and $k = \binom{n}{2}$. Clearly $|\langle G, s, t, 1^m, \alpha, \beta\rangle|$ is bounded polynomially in $n$ and $m$. If $f$ is a logspace-computable function that reduces $B$ to $S$, the bound on the length of queries made by $f$ on inputs of length $|\langle G, s, t, 1^m, \alpha, \beta\rangle|$ is some polynomial $q(n, m)$. Let $p(n, m)$ be a polynomial that bounds the number of strings in $S$ of length at most $q(n, m)$. We will choose the smallest $m$ of the form $2 \cdot 3^\ell$ such that $2^m / p(n, m) \ge k = \binom{n}{2}$. It is clear that $m = O(\log n)$. Let $\mathbb{F}$ denote the finite extension $GF(2^m)$ of $GF(2)$.

Our parallel algorithm for *DAG-STCON* begins by computing $f(\langle G, s, t, 1^m, \alpha, \beta\rangle)$ for all $\alpha, \beta \in \mathbb{F}$. (Setting up the required powers of $\alpha$ is an easily accomplished task, since it can be precomputed off-line in logspace.) For every $\alpha \in \mathbb{F}$, there is a unique element $\beta_\alpha \in \mathbb{F}$ such that $\langle G, s, t, 1^m, \alpha, \beta_\alpha\rangle \in B$, and therefore $f$ maps precisely one tuple of the form $\langle G, s, t, 1^m, \alpha, \beta\rangle$ into $S$. Since $2^m / p(n, m) \ge k$, there is at least one string $w^* \in S$ such that the number of $\alpha$ satisfying $f(\langle G, s, t, 1^m, \alpha, \beta_\alpha\rangle) = w^*$ is at least $k$. As before, strings $w$ that have $\ge k$ pre-images $\alpha$ will be called *popular*.

For any $w$, whenever $f(\langle G, s, t, 1^m, \alpha, \beta\rangle) = w$, under the *assumption* that $w \in S$ we have an equation

$$1\, X_{u_0 v_0} + \alpha X_{u_1 v_1} + \alpha^2 X_{u_2 v_2} + \ldots + \alpha^{k-1} X_{u_{k-1} v_{k-1}} = \beta$$

in the variables $X_{uv}$. Thus for every *popular* $w$, we will have a *system* of at least $k$ such equations; moreover, the system of equations is *correct* if and only if $w \in S$. Of course, there could be many popular $w$, and we don't know which ones are in $S$. To handle this, we will *assume* that every popular $w$ is a string in $S$, and attempt to solve for the $X_{uv}$'s for all $u, v \in V, u < v$. This scheme produces a polynomial number of sets of solutions. As long as there is at least one popular $w^* \in S$, one of our assumptions must be correct, and we will have the correct solution. It remains, therefore, to show: (1) how to *solve* the systems of equations by a poly-size log-depth circuit, and (2) how to *verify* the correctness of solutions.

*Solution of the Systems of Equations.*

For every popular $w$, when the equations produced are written as matrix-vector product of the form $\mathcal{A}X = \mathcal{B}$, the $k \times k$ matrix $\mathcal{A}$ obtained is a *Vandermonde* matrix. Moreover, since the $\alpha$'s are distinct, the matrix $\mathcal{A}$ has full rank over $\mathbb{F}$. By lemma 2.11, this system of equations can be solved in $\mathrm{NC}^1$.

*Verification of the Solutions.*

Each set of solutions for the $X_{uv}$'s can be assumed to be in the form of a matrix $X$ of the same size as $A_G$, the adjacency matrix of $G$. By its definition, $X_{uv}$ is 1 only if $(u, v) \in E$. Therefore, for each candidate set of solutions, we will first verify that the condition $X_{uv} \leq A_G(u, v)$ holds for all $u, v \in V$. It is easy to see that this test can be performed simultaneously on all sets of solutions by a polynomial-size, log-depth circuit. Note that since $A_G$ is an upper-triangular matrix with zero diagonal, every $X$ that passes this test is also upper-triangular matrix with zero diagonal.

For every $u \in V$, we first compute the boolean variable $X_u$ that is 1 if and only if $u = t$ or there exists some $v$ such that $X_{uv} = 1$. In matrix terms, $X_u = 1$ if and only if $u = t$ or there is at least one 1 in the row corresponding to $u$ in the matrix $X$. This computation can be easily done by a polynomial-size, log-depth circuit, since it only requires computing the OR of $n$ bits. Next we perform the following *local consistency test*: for every $u$ such that $X_u = 1$ and for every $v$ such that $X_{uv} = 1$, verify that $X_v = 1$. This test ensures that if $X$ promises a path from $u$ to $t$ with $v$ as the first vertex, then indeed $X$ also promises some path from $v$ to $t$. Notice that the latter path cannot include the vertex $u$ since $G$ is acyclic. This is important because, otherwise, it is possible that $X$ passes this test by setting $X_{uv} = X_{vu} = 1$ even though $t$ is reachable from neither of $u$ and $v$. It is clear that the local consistency test can be performed by a polynomial-size log-depth circuit.

Finally we argue that there exists a path from $s$ to $t$ if and only if some set of solutions $X$ which passes all these tests and has $X_s = 1$. Clearly, if there is a path from $s$ to $t$ then the correct solution for $X$ will pass all the tests and have $X_s = 1$.

Next we claim that if $X$ passes all the tests, then $X_z = 0$ for all $z > t$. The claim is vacuous if $t$ is the last vertex. Otherwise, we prove the claim by induction, starting from the last vertex. By the first test against the adjacency matrix and since the graph is layered, the base case is clear. Assume inductively for some $t < z_0$, that $X_z = 0$ for all $z$ such that $z_0 < z$. If $X_{z_0} = 1$, then by the definition of $X_{z_0}$, for some $z > z_0$, $X_{z_0 z} = 1$. However, it then fails the *local consistency test*, since $X_z = 0$.

It follows in particular that if $s > t$ and $X$ passes all the tests, then $X_s = 0$.

To complete the proof, suppose that $X$ passes all the tests. We argue that whenever $X_u = 1$ for some $u \leq t$, there is a path from $u$ to $t$. The base case, namely $u = t$, is trivial. Suppose $X_u = 1$ for some $u < t$. By definition, there is a vertex $v$ such that $X_{uv} = 1$ and $X_v = 1$. The first test ensures that there is an edge $(u, v)$. If $v = t$, it is clear that there is a path from $u$ to $t$. If $v < t$, by the inductive hypothesis, there is a path from $v$ to $t$, which, together with the edge $(u, v)$, gives a path from $u$ to $t$. $\square$

**Corollary 2.21 (Theorem 2.4)** *There is a sparse hard set for* NL *under logspace many-one reductions iff* NL = L.

**Corollary 2.22** *There is a sparse hard set for* NL *under many-one reductions computable in logspace-uniform* $\mathrm{NC}^1$ *iff* NL = *logspace-uniform* $\mathrm{NC}^1$.

The proofs of the next two theorem combines the above technique with ideas from Section 2.5.

**Theorem 2.23** *If there is a sparse hard set for* NL *under logspace-computable randomized many-one reductions with two-sided error, then* $\mathrm{NL} = \mathrm{RL}_2$, *where* $\mathrm{RL}_2$ *is the class of languages accepted by logspace Turing machines with two-way access to the random tape. If there is a sparse hard set for* NL *under randomized many-one reductions with two-sided error that are computable in logspace-uniform* $\mathrm{NC}^1$*, then* $\mathrm{NL} \subseteq \mathrm{RNC}^1$.

## 2.7 Summary and Further Results

In this chapter, we considered two 20-year old conjectures of Hartmanis concerning the existence of sparse hard sets for the complexity classes P and NL. Our main results of this chapter affirm both conjectures. We conclude this chapter with a few remarks about the proof methodology, and about some related developments.

Our proof of the conjecture of Hartmanis for P progressed in three stages, beginning with an innovative idea of Ogihara [Ogi95]. Our first point of deviation from Ogihara's proof of a weaker result was our introduction of *randomization* into the picture. This twist, combined with a linear-algebraic approach, immediately connected the problem to *parallel complexity*, and resulted in a somewhat better result, namely the collapse $\mathrm{P} \subseteq \mathrm{RNC}^2$ under the assumption about the existence of a sparse hard set for P.

The approach using probabilistic and algebraic ideas quickly led to the development of further techniques and sharper results. Since the randomized algorithm of Section 2.2 had a strong algebraic flavor, the question of how to derandomize this could be phrased in clear linear-algebraic terms, which further led to the developments in Section 2.3. This is where our next twist came in, namely the use of finite field theory in the construction of a small sample space to derandomize the algorithm of Section 2.2.

The result collapsing P to $\mathrm{NC}^2$ does not settle the conjecture of Hartmanis, which is about many-one reductions computable in logarithmic space. The bottleneck is that the best-known algorithm for solving a system of equations over $GF(2)$ gives only an $\mathrm{NC}^2$ simulation. However, the definition of the sample space $D$ led us to a system of equations over a finite extension $\mathbb{F}$ of $GF(2)$ with a

*Vandermonde* coefficient matrix. We then solve this system using a *closed formula* that could be implemented in NC$^1$, through the application of the Discrete Fourier Transform over $\mathbb{F}$.

A curious aspect of this three-stage proof is that we could have instead applied the ideas of Section 2.5 directly to the original problem, and shown that if P has a sparse hard set under NC$^1$ reductions, then P $\subseteq$ RNC$^1$. If we had taken that path instead of the linear-algebraic approach of setting up a system of equations and solving them, we probably wouldn't have made further progress—there doesn't seem to be any easy way of derandomizing the Goldreich-Levin algorithm!

It is fair to say that every twist that we made—the introduction of randomization, setting up equations and solving them, derandomization through the use of a small sample space, setting up the Vandermonde system, and its solution using the DFT—was crucial in the eventual resolution of the conjecture of Hartmanis. It is inconceivable, at least to this author, that we could have directly arrived at the final solution without going through the algorithms of Sections 2.2 and 2.3.

The proof of the resolution of the sparse set conjecture for NL brings to fore yet another advantage of the randomization-derandomization approach. In the proof of Theorem 2.19, we needed to prove that the auxiliary language $B$ is in NL. To accomplish this, we appealed to many special properties of the field $\mathbb{Z}_2[X]/(X^{2\cdot 3^\ell}+X^{3^\ell}+1)$, such as the fact that the "mod" operation with respect to the polynomial $X^{2\cdot 3^\ell}+X^{3^\ell}+1$ can be accomplished by simple shifts and additions, and the fact that we can work out precise closed-form expressions for each bit of the product. This turned out to be crucial when computing products during a space-bounded computation. It is not clear whether the NL result could be proved by working in a field such as $\mathbb{Z}/p$ for an appropriate prime number $p$.

In [CNS96], we consider the case of sparse hard sets under bounded truth-table reducibility, which are a slight generalization of many-one reducibility, although still far more restricted than general Turing reducibility.

**Definition 2.9 (Bounded Truth-Table Reduction)** *A language $A$ is said to be reducible to $B$ by a $b$-**tt reduction** $f$ if for every $x$, $f$ outputs a list of $b$ queries $w_1,\ldots,w_b$, and a boolean predicate $e$ on $b$ variables, such that $x \in A$ if and only if $e(B(w_1),\ldots,B(w_b))$.*

Under the assumption that P has a sparse hard set under logspace computable bounded truth-table reductions, we first show that P $\subseteq$ RNC$^2$. The proof of this result relies on solving a system of equations where a fraction of the equations could be erroneous. We then derandomize this algorithm, taking advantage of certain *error-correcting* capabilities of the set $D$ used before for the many-one case. The set $D$ is formally identical to an *error-correcting code* [vL91] that is obtained by concatenating a certain Reed-Solomon code with a Hadamard code.

It is an indication of the effectiveness of algebraic and derandomization techniques that the proof for the many-one case can be generalized to account for the case of bounded truth-table reductions. We note that in the NP case it took the research community 10 years to generalize Mahaney's result for many-one reducibility to Ogihara-Watanabe's theorem for bounded truth-table reducibility [OW91]. Our result for bounded truth-table reductions has been further strengthened by van Melkebeek [Mel96], who showed that if P has a sparse hard set under logspace-computable bounded truth-table reductions, then P $=$ LOGSPACE.

As a final retrospective remark, we note that when the conjecture of Hartmanis was made in 1978, probabilistic algorithms had not gained popularity, the use of finite field theory in theoretical computer science was relatively unknown, and the parallel complexity classes $\mathrm{NC}^k$ had not even been defined! Thus, in a sense, the results of this chapter rely on purely "modern" techniques in theoretical computer science to resolve an old conjecture in complexity theory.

Finally, as a useful by-product of our work on *quasipolynomially* sparse sets [CNS96], we have developed deterministic $O(\log^2 n \log\log n)$ time and randomized $O(\log^2 n)$ time parallel algorithms to compute the rank of of an $m \times n$ matrix over $GF(2)$, where $m = n^{O(\log n)}$. This work draws on earlier work by Mulmuley [Mul87] and by Chistov [Chi85], and improves the best known previous algorithm for this problem.

# Chapter 3

# Measure Theory and Pseudorandomness

Classical measure theory concerns itself with functions that assign, to each subset of a set $S$, a real number, called the *measure*, that captures the "size" of the subset. The intuitive idea is that small subsets of $S$ should be assigned small values and large subsets of $S$ should be assigned large values.

Perhaps the simplest example is that of assigning a measure to each subset of the set $\mathbb{R}$ of real numbers. To each subset $S \subseteq \mathbb{R}$, let $\mu(S)$ be the sum of the lengths of open real intervals that cover $S$. It is clear that the measure of $\mathbb{R}$ itself is $\infty$. What about finite sets? Clearly every finite set can be covered by open intervals, the sum of whose lengths can be made arbitrarily small; this defines the notion of *zero* measure. Not surprisingly, any countable subset of $\mathbb{R}$ also has measure zero: given any $\varepsilon > 0$, cover the $n$-th point in the set by an open interval of size $6\varepsilon/(\pi^2 n^2)$[1]. It is almost immediate that the collection of sets of real numbers of measure zero is closed under finite unions, and in fact, under countable unions.

The development of classical measure theory has been a subject of central focus in two central branches of mathematics, namely analysis and topology. More recently, in the twentieth century, measure theory has contributed significantly to the development of modern set theory and has put probability theory on a firm mathematical footing. In turn, developments in modern set theory have had a rather striking effect on measure theory, exemplified by the discovery of the existence of non-measurable sets through the use of the Axiom of Choice.

Given the tremendous impact that classical measure theory has had in mathematics, it is natural to ask whether measure concepts may have a substantial impact on complexity theory. It is clear that we may view any language $A$ over the alphabet $\Sigma$ as a real number $r_A \in [0,1]$ by viewing its characteristic sequence $\chi_A$ as a real number written in binary notation. Therefore we may define the measure of a complexity class $\mathcal{C}$ as the measure of the subset $\{r_A \mid A \in \mathcal{C}\}$. However, we are immediately faced with a stumbling block: all complexity classes that are defined as a collection of languages accepted by some set of Turing machines are countable, and have measure zero in the space of all languages. In other words, studying the classical measure of complexity classes may not yield any new insight into their structure.

[1] $\sum_{n=1}^{\infty} = \pi^2/6$

## 3.1 Resource-Bounded Measure

Since all our favorite complexity classes, namely P, NP, PSPACE, EXP, etc. have measure zero, it is hopeless to pursue a classical measure-theoretic investigation of these classes as a means to prove separations among these. However, if we are trying to address, for example, the P vs. NP question, we may still ask the question: *what is the measure of these classes as subsets of a larger class such as* EXP *that contains these?* In other words, we may ask if we can restrict our attention to a reasonably large complexity class such as EXP and define a measure on (the subsets of) this class. Jack Lutz [Lut92a] raised precisely this question, and has set up a solid mathematical foundation for a theory of measure within complexity classes.

### 3.1.1 A Primer on Resource-Bounded Measure

Lutz's theory was originally developed along classical lines, via complexity-bounded *null covers*. However, this was proved equivalent to a formulation using *martingales* that is much simpler to describe. We begin by reviewing the definition of classical measure zero in this formulation.

Via binary expansions, we first identify $[0,1]$ with the infinite complete binary tree $\Omega$. Second, a language $A \subseteq \Sigma^*$ is identified with its characteristic sequence $\chi_A$ under the fixed lexicographic ordering $\prec$ of all strings in $\Sigma^*$. The sequence $\chi_A$ is then identified with an infinite path in the tree $\Omega$. Let $\Psi$ denote the set of all downward paths of finite length that originate at the root of $\Omega$. It is clear that $\Psi$ is in one-to-one correspondence with $\{0,1\}^*$, the set of all possible prefixes of characteristic sequences of languages over $\Sigma$. We will consistently refer to members of $\Psi$ by their labels in $\{0,1\}^*$.

**Definition 3.1 (Martingale)** *A* ***martingale*** *is a function $\kappa$ from $\Psi$ into the nonnegative reals such that $\kappa(\lambda) = 1$, and such that the the following* exact average law *is satisfied: for all $w \in \Psi$,*

$$\kappa(w) = \frac{\kappa(w0) + \kappa(w1)}{2}. \tag{3.1}$$

The requirement that $\kappa(\lambda) = 1$ is, of course, only cosmetic; it can be chosen to be any finite value. However, keeping $\kappa(\lambda) = 1$ simplifies the exposition, particularly when constructing a new martingale as the weighted sum of some martingales.

The *success set* of a martingale $\kappa$ is defined to be

$$S^\infty[\kappa] = \{A : \lim_{w \sqsubseteq A} \kappa(w) = +\infty\}.$$

We say the martingale *succeeds* on $\chi_A$ iff $A \in S^\infty[\kappa]$, and we say the martingale succeeds on a set $\mathcal{C}$ (or *covers* $\mathcal{C}$) if $\mathcal{C} \subseteq S^\infty[\kappa]$. The key observation now is the following:

**Proposition 3.1** *A set $\mathcal{C}$ has classical measure zero iff there is a martingale that succeeds on $\mathcal{C}$.*

The proof from left to right constructs a martingale from a collection of open intervals that cover $\mathcal{C}$. The proof of the other direction, of course, is just Markov's inequality, but offers more insight and intuition about martingales and the notion of measure zero, which we sketch briefly.

A martingale $\kappa$ that covers $S^\infty[\kappa]$ may be thought of a proof of the fact that the probability that a random language belongs to $S^\infty[\kappa]$ is zero. With this in mind, we will view a martingale as a betting strategy working on the tree $\Omega$. It starts with a capital of \$1 at the root. At any node $w \in \Psi$, the martingale has a total capital $\kappa(w)$. The node $w$ encodes the memberships of the $|w|$ words preceding some string $x \in \Sigma^*$. (If we write the length of $w$ in binary, and write $|w|+1 = 1x$, then the lexicographically next string is $x$.)

We pause to define a few quantities that facilitate discussion:

(1) The *bet amount* $\beta(w) \in [-2\kappa(w), 2\kappa(w)]$, defined by $\beta(w) \doteq \kappa(w1) - \kappa(w0)$.

(2) The *proportional bet* $b(w) \in [-1,1]$, defined by

$$b(w) = \beta(w)/(2\kappa(w)) = (\kappa(w1) - \kappa(w))/\kappa(w)$$

, if $\kappa(w) \neq 0$.

(3) The conditional probabilities $p_0(w), p_1(w) \in [0,1]$, defined by

$$p_1(w) \doteq \frac{\kappa(w1)}{2\kappa(w)} = \frac{1+b(w)}{2}; \qquad p_0(w) \doteq \frac{\kappa(w0)}{2\kappa(w)} = \frac{1-b(w)}{2}.$$

The bet satisfies $|\beta(w)| \leq \kappa(w)$, and the amount $\beta(w)$ is wagered on the assertion that for $A \in S^\infty[\kappa]$, $x$ belongs to $A$. If $\beta(w) \geq 0$, then the martingale bets that $x$ belongs to $A \in S^\infty[\kappa]$, and if $\beta(w) \leq 0$, then the martingale bets that $x$ does not belong to $A \in S^\infty[\kappa]$. The magnitude $|\beta(w)|$ of the bet reflects the martingale's confidence in its assertion. Another viewpoint is that the martingale asserts that a $p_1(w)$ fraction of all languages $A$ with $w \sqsubseteq \chi_A$ belong to $S^\infty[\kappa]$.

Now let $A$ be an arbitrary language (not necessarily in $S^\infty[\kappa]$), let $w \sqsubseteq \chi_A$, and let $x = x(w)$ denote the lexicograhically next string on which the bet is placed. We say that the martingale *wins* its bet on $x$ (with respect to $A$) and makes a *profit* of $\beta(w)$ if either $x \in A$ and $\beta(w) > 0$, or if $x \notin A$ and $\beta(w) < 0$. Otherwise, we say the martingale *loses* its bet on $x$ (again with respect to $A$).

As a gentle reminder to the reader, we note that the notion of winning or losing any given bet is meaningful only with respect to some language $A$; there is no absolute notion of a bet being successful. The crucial point, of course, is that if the martingale succeeds on some language $A$, then since $\lim_{w \sqsubseteq \chi_A} \kappa(w) = +\infty$, the martingale must have won infinitely many of its bets *along* $\chi_A$, and not only that, it must have cumulatively amassed the winnings of infinitely many bets.

What does it take for a martingale to succeed on a class $\mathcal{C}$ of languages? Clearly, it must make infinitely many successful bets along $\chi_A$ for every $A \in \mathcal{C}$. Very intuitively, this is saying that the martingale possesses knowledge about some property of all languages in $\mathcal{C}$, and is in fact, capable of predicting the memberships in the languages in $A \in \mathcal{C}$, for the bets are made by looking only at prefixes of $\chi_A$. As a simple example, it is very easy to cover the class of all sparse languages: the martingale always bets 99% of its current capital that the next string will be "out," and bets 1% of its current capital that the next string will be "in." Of course, the martingale doesn't have any one language "in mind," but is trying to collectively capture the class of all sparse languages by exploiting the property that they all have very few strings.

We have only introduced what it means for a class of languages to have measure zero. Next we will describe how this formulation helps us define a measure on complexity clases. Lutz's idea is to place various complexity restrictions on the computation of the martingale. In particular, to define a measure on deterministic time complexity classes, the running time of an algorithm that computes the martingale function will be limited. The time to compute the martingale is measured in terms of the length of the description of the node $w \in \Psi$ (which is a prefix of the characteristic sequences of all languages below the node $w$, i.e. an open interval in $\mathbb{R}$ when viewed as a finite binary sequence). In other words, the idea to generalize classical measure is to consider the complexity of the decision problem: Given the description of an open interval $I$ and some $\varepsilon > 0$, decide whether or not $I$ is included in the cover whose total length is $\varepsilon$.

For $w \in \Psi$, let $x = x(w)$ denote the lexicographically "next string," given by $1x = |w| + 1$. Lutz has shown [Lut92a] that by considering only martingales running in time polynomial (resp. quasipolynomial) in the length of $w$, i.e. in time $2^{O(|x|)}$ (resp. $2^{|x|^{O(1)}}$), one gets a notion of measure at exponential time E (resp. EXP), and that this is true also of other "nice" complexity classes at least as large as E. Formally, if the class $\mathcal{D}$ is defined by a collection $\mathcal{R}$ of resource bounds that is closed under squaring, then Lutz defines $\Delta(\mathcal{D})$ to be the class of martingales computable within the bound $r(\log N)$ for some function $r(N) \in \mathcal{R}$.

**Definition 3.2 (Measure Zero)** *For any class $\mathcal{C}$, we write $\mu_{\Delta(\mathcal{D})}(\mathcal{C}) = 0$, and say $\mathcal{C}$* ***has** $\Delta(\mathcal{D})$-**measure zero**, if there is a martingale computable in $\Delta(\mathcal{D})$ that succeeds on $\mathcal{C}$. We write $\mu(\mathcal{C}|\mathcal{D}) = 0$, and say $\mathcal{C}$ **has measure zero in** $\mathcal{D}$, if $\mu_{\Delta(\mathcal{D})}(\mathcal{C} \cap \mathcal{D}) = 0$. Also, we say $\mathcal{C}$ **has measure one in** $\mathcal{D}$ if $\mu_{\Delta(\mathcal{D})}(\mathcal{D} - \mathcal{C}) = 0$.*

The classical Kolmogorov zero-one law [Oxt80] dictates that classes closed under finite variations have measure zero or measure one, if they are measurable at all. Mayordomo [May94], demonstrated that the Kolmogorov zero-one law applies to the $\Delta(\mathcal{D})$ measures as well. Since the complexity classes whose measure we are interested in are closed under finite variations, it is sufficient to define measure zero (and measure one). Moreover, Mayordomo [May94] also demonstrated that in these cases, the definition of $\mu(\mathcal{C}|\mathcal{D}) = 0$ is robust under certain changes to the definition of measure zero, most notably under relaxing (3.1) to the inequality $\kappa(w) \geq (\kappa(w0) + \kappa(w1))/2$, and under relaxing the limit condition in the definition of success to a limsup.

Lutz [Lut92a] argues the above notion of measure on complexity classes is *reasonable* for the following reasons, besides the fact that this is a technically sound generalization of classical measure:

(1) Clases that are evidently small (e.g. the singletons $\{A\}$ for some $A \in \mathcal{D}$) have measure zero.

(2) Finite unions, as well as certain resource-bounded countable unions, of measure zero classes have measure zero.

(3) (*Measure Conservation*) The class $\mathcal{D}$ itself does not have $\Delta(\mathcal{D})$-measure zero.

Condition (3) in the above is extremely important in applications of resource-bounded measure theory, for it says that if we prove $\mu_{\Delta(\mathcal{D})}(\mathcal{C}) = 0$, then we would have *minimally* proved that $\mathcal{C} \neq \mathcal{D}$. This is also an instance of the *probabilistic method* [AS92a] where, in order to prove that there is a language in $\mathcal{D}$ that is not in $\mathcal{C}$, we demonstrate that almost every language in $\mathcal{D}$ (i.e. a measure one subset of $\mathcal{D}$) is not in $\mathcal{C}$. To substantiate Condition (1) above, Lutz shows that the classical time-hierarchy theorems [HS65] carry over to resource-bounded measure. In particular, the class P, and in fact, DTIME$[2^{cn}]$ for any fixed $c$, has $\Delta(\mathrm{E})$-measure zero, and E itself, as well as DTIME$[2^{n^c}]$ for any fixed $c$, has $\Delta(\mathrm{EXP})$-measure zero.

## 3.1.2 The Hypothesis $\mu(\mathrm{NP}|\mathrm{EXP}) \neq 0$

The fact that the deterministic time-hierarchy theorem carries over in the stronger measure setting naturally suggests that considerable insight might be gained about the structure of complexity classes by studying the measure (in a suitably large class like EXP) of natural and interesting classes such as NP, $\Sigma_2^p$, PH PSPACE, BPP, P/poly, etc. that are not defined as a collection of languages accepted by time-bounded deterministic Turing machines. Of course, such an endeavor must be necessarily difficult, because proving any statement about the measure of NP, for example, would immediately settle at least one major open question in complexity theory. If NP has measure zero in EXP, then NP $\neq$ EXP; if NP does not have measure zero in EXP, then P $\neq$ NP.

After all, what does the statement $\mu(\mathrm{NP}|\mathrm{EXP}) = 0$ mean? It asserts the existence of a single exponential-time algorithm that predicts every NP language reasonably well. However, there are NP languages whose search spaces have arbitrarily large exponential sizes. With the view that no single exponential-time algorithm can gain significant advantage on all such languages, Lutz has advanced the hypothesis that $\mu(\mathrm{NP}|\mathrm{EXP}) \neq 0$, that is, either NP has measure one in EXP, or is not measurable at all.

This hypothesis is clearly as strong as P $\neq$ NP, and possibly much stronger: there are a wide variety of complexity-theoretic consequences of the hypothesis $\mu(\mathrm{NP}|\mathrm{EXP}) \neq 0$ that are not known to follow from the hypothesis P $\neq$ NP. For example, the hypothesis implies the following: NP has P-bi-immune sets [May92]; for every $c > 0$, there are languages in NP that require deterministic time more than $2^{n^c}$; there would be NP-complete sets under Turing (Cook) reductions that are not complete under many-one (Karp) reductions [LM94]; BPP $\subseteq \mathrm{P}^{\mathrm{NP}}$ [AS94a]; the NP-complete satisfiability problem would, in fact, be hard for a measure one subset of EXP [JL93]. Thus, Lutz [Lut93] argues, we should study the hypothesis $\mu(\mathrm{NP}|\mathrm{EXP}) \neq 0$ for its explanatory power, for the wealth of consequences that follow from this hypothesis. (And who knows, this hypothesis or its negation may turn out to be easier to prove than P $\neq$ NP or NP $\neq$ EXP—after all, there are plenty of instances in mathematics where it is often easier to prove a theorem by proving a stronger assertion!)

## 3.1.3 This Chapter: The Measure of P/poly

While the central question of interest in the measure-theoretic framework is that of the measure of NP in EXP, not much is known about the measure in EXP of any complexity class that is not defined by time-bounded deterministic Turing machines. For example, nothing is known about one-sided error randomized polynomial time (RP), which is a subset of NP, or about the measure of P/poly, the class of languages with polynomial-sized circuits. In the preamble for Chapter 2, we discussed the significance of this class in complexity theory. Truly, P/poly remains a mysterious class, at once believed to be much weaker than NP, and at the same time believed to be quite powerful because of the nonuniformity.

Lutz [Lut92a] proved that P/poly has $\Delta(\mathrm{E}_3)$-measure zero, where $\mathrm{E}_3$ denotes the slightly super-exponential class $\mathrm{E}_3 = \mathrm{DTIME}[2^{2^{\log^{O(1)} n}}]$, and in fact, has $\Delta(\mathrm{ESPACE})$-measure zero. This was followed in [May94], where Mayordomo showed that P/poly has measure zero in the third level of the alternating exponential-time hierarchy, using the approximate counting of [Sto77]. All these results were proved in the more or less straightforward way of enumerating all circuits and simulating them (just as one would prove that P has measure zero in EXP). As a benefit of doing so, these results are absolute in the sense that they don't rely on any unproven hypothesis, and also in the sense that they do not just prove that P/poly has measure zero *in* ESPACE or EH, but rather demonstrate the existence of a $\Delta(\mathrm{ESPACE})$ or $\Delta(\mathrm{EH})$ martingale that succeeds on all of P/poly, including all the languages that have a strictly nonuniform family of circuits.

Prior to our paper [RSC95], the question of the measure of P/poly at the exponential-time level ($\mu_{\Delta(\mathrm{EXP})}(\mathrm{P/poly})$ or $\mu(\mathrm{P/poly}|\mathrm{EXP})$) was completely unanswered. Since there are oracles that make $\mathrm{EXP} = \mathrm{P/poly}$ [Hel86], it was believed to be difficult to show that $\mu(\mathrm{P/poly}|\mathrm{EXP}) = 0$.

The research summarized in this chapter is described in [RSC95], and establishes a connection between the measure of P/poly and the existence of strong pseudorandom generators. Our main result in this chapter is that if there are strong pseudorandom generators, then P/poly does not have measure zero in EXP. In fact, we strengthen it further, and show under the same hypothesis that P/poly is not measurable at all in EXP! Evidently, our result does not use the ideas that were used to construct the oracle that makes $\mathrm{EXP} = \mathrm{P/poly}$, since relative to that oracle, P/poly has measure one in EXP.

Our proof technique is inspired by the recent work of Razborov and Rudich [RR94], who introduced the notion of a *natural proof*, and established a connection between the existence of strong pseudorandom generators and the limitations of natural proofs to prove superpolynomial lower bounds for explicit problems. Specifically, they showed that if there is a strong pseudorandom generator, then there is no "P/poly natural proof against P/poly." We show that a martingale that witnesses measure-zero for $\mathrm{P/poly} \cap \mathrm{EXP}$ yields a natural proof of slightly higher complexity, but one that still suffices to refute the existence of strong pseudorandom generators.

OUTLINE OF THIS CHAPTER. In the next section, we introduce the notion of a pseudorandom generator, and state a technical theorem that shows how to increase the "streching factor" while preserving the "hardness" of the generator. In Section 3.3, we prove our main theorem. Although our technique was inspired by the idea of a natural proof, we provide a simple and direct proof of our main theorem, completely bypassing the notion of a natural proof. The reader interested only in the main theorem (and not in the connection to natural proofs) can find a self-contained proof of the main theorem here. In Section 3.4, we strengthen our main theorem and show that under the assumption of strong pseudorandom generators, P/poly is not measurable at all in EXP. The technical idea of this section is of independent interest, and it answers an old open question in resource-bounded measure theory: under what condition does NP not have measure *one* in EXP? In Section 3.5, we show the connection to natural proofs. We introduce the framework of the natural proofs, and provide a quick sketch of the main theorem of Razborov and Rudich [RR94], and demonstrate how a martingale yields a natural proof. In Section 3.6, we prove a version of our main theorem that works under a weaker hypothesis about pseudorandom generators. In Section 3.7, we provide a partial converse of our connection between measure theory and natural proofs, and finally in Section 3.8, we make some concluding remarks.

## 3.2 Pseudorandom Generators

A pseudorandom generator is formally a sequence $\{G_n\}$, where each $G_n$ is a function from $\{0,1\}^n$ to $\{0,1\}^{\ell(n)}$, and $\ell(n) > n$. Intuitively, $G_n$ is designed to "stretch" a sequence of $n$ truly random bits into a longer sequence of bits that appear random to resource-bounded adversaries.

**Definition 3.3 (Hardness of Pseudorandom Generator)** *Let $G = \{G_n\}$ be a pseudorandom generator. The* hardness of $G$ at $n$, $H(G_n)$, *is defined to be the largest integer $S(n)$ such that for every $\ell(n)$-input circuit $C$ of size at most $S(n)$,*

$$\left| \Pr_{y\in\{0,1\}^{\ell(n)}}[C(y)=1] - \Pr_{x\in\{0,1\}^n}[C(G_n(x))=1] \right| \leq \frac{1}{S(n)}.$$

*$G$ is said to be of* hardness at least $h(\cdot)$ *if for all but finitely many $n$, $H(G_n) \geq h(n)$.*

To simplify matters, we will assume that pseudorandom generators stretch $n$ bits to $2n$ bits. This is not much of a technical concern, since so long as $\ell(n) = n^{O(1)}$, $H(G_n)$ is invariant up to constant factors (for a proof, see, for example, [BH89, HILL91]). The next proposition, however, shows that if we have a generator of hardness $2^{n^{\Omega(1)}}$ that stretches $n$ bits to $2n$ bits, then we can in fact build a generator that stretches $n^{O(1)}$ bits to $2^n$ bits. Moreover, the pseudorandom generator thus constructed permits efficient computation.

**Proposition 3.2 ([RR94], after [GGM86])** *Suppose there is a pseudorandom generator $G = \{G_k : \{0,1\}^k \to \{0,1\}^{2k}\}$ of hardness $2^{k^\varepsilon}$ for some $\varepsilon > 0$. Then for any $c$, there is a pseudorandom generator $\widetilde{G} = \{\widetilde{G}_n : \{0,1\}^{n^c} \to \{0,1\}^{2^n}\}$ of hardness $2^{n^c}$. Moreover, suppose $G$ is computable by a family of polynomial-size circuits. Then there is a family of circuits $\{C_n\}$ of size $n^{O(c/\varepsilon)}$ such that for every $n$ and $x \in \{0,1\}^{n^c}$ and every $y \in \{0,1\}^{2^n}$, $C_n(x,y)$ outputs the $y$-th bit of $\widetilde{G}_n(x)$.*

**Proof.** For any $n$, set $k = n^{c/\varepsilon}$. For $x \in \{0,1\}^k$, let $G_k^0(x)$ denote the first $k$ bits of $G_k(x)$, and let $G_k^1(x)$ denote the last $k$ bits of $G_k(x)$. Also, for $y \in \{0,1\}^n$, define the $y$-th bit of $\widetilde{G}_n(x)$ to be the first bit of $G_k^{y_n} \circ G_k^{y_{n-1}} \circ \ldots \circ G_k^{y_1}(x)$. It is clear that there is a circuit of size $n^{O(c/\varepsilon)}$ that computes $\widetilde{G}_n(y)$ for all $y \in \{0,1\}^n$.

Suppose by way of contradiction there is a circuit $C_n$ of size $2^{O(n^c)}$ that achieves a bias of $2^{-O(n^c)}$ in distinguishing between $G_k(x)$ when $x$ is chosen randomly from $\{0,1\}^k$ and a randomly chosen $2^n$-bit string. We will show that there is a circuit $D_k$ of size $2^{O(n^c)} = 2^{k^\varepsilon}$ that achieves a bias of $2^{-O(n^c)} = 2^{-k^\varepsilon}$ in distinguishing between $G(x)$ when $x$ is chosen randomly from $\{0,1\}^k$, and a randomly chosen $2k$-bit string.

Consider the full binary tree $T$ of height $n$. Label the internal nodes of $T$ by $v_1, v_2, \ldots, v_{2^n-1}$ such that if $v_i$ is a child of $v_j$ then $i < j$. Note that $T$ has $2^n$ leaves; we will associate the leaves in one-to-one correspondence with all strings of length $n$. Denote by $T_i$ the union of subtrees of $T$ consisting of the nodes $v_1, \ldots, v_i$, together with all leaves. For a leaf $y$ of $T$ let $v_i(y)$ be the root of the subtree in $T_i$ containing $y$. For all leaves $y$, define $G_{0,y}$ to be the identity function, and let $G_{i,y}$ denote the composition $G_{y_n} \circ G_{y_{n-1}} \cdots G_{y_{n-h(i,y)+1}}$. Here $h(i,y)$ denotes the height of $y$ in $T_i$, or the distance between $v_i(y)$ and $y$. To each internal node $v$ of the tree $T$, assign a string $x_v$ chosen uniformly at random from $\{0,1\}^k$. Next, define the random collection $f_i$ to be the collection of functions $\{f_{i,x}\}$ described as follows. Let $z$ be a leaf of the tree. Define $f_{i,x}(z)$ to be the first bit of $G_{i,z}(x_{v_i(z)})$. Note that $f_0$ is just a random boolean function on $n$ variables, and $f_{2^n-1}$ is just $\widetilde{G}_n(x)$ defined above. We know that

$$|\Pr[C_n(f_0)=1]-\Pr[C_n(f_x)=1]| \geq 2^{-O(n^c)}.$$

Therefore, there must exist an index $i$ such that

$$|\Pr[C_n(f_i)=1]-\Pr[C_n(f_{i+1})=1]| \geq 2^{-O(n^c)}.$$

At this point, an averaging argument shows that we can fix all the random strings assigned to the nodes of $T$ except the children of $v_{i+1}$ while preserving the bias. (This might determine many of the bits of $f_x$.) Now there are two ways of assigning strings to the children of $v_{i+1}$: either assign them both independently chosen random strings from $\{0,1\}^k$, or assign a random string $u$ to $v_{i+1}$ and assign to its two children the strings $G_k^0(u)$ and $G_k^1(u)$ respectively. The crucial observation we make is that if these two nodes are assigned strings in the first way, then the resulting boolean function induced on the leaves is precisely $f_i$, and if they are assigned strings in the second way, then the resulting boolean function induced on the leaves is precisely $f_{i+1}$. To complete the proof, we will build a circuit $D_n$ that takes a string in $\{0,1\}^{2k}$ and computes the resulting boolean function at the leaves (which one of $f_i$ or $f_{i+1}$) as described, and feeds the result ($f_i$ or $f_{i+1}$) to $C_n$. Note that computing $f_i$ or $f_{i+1}$ can be done in time $O(2^n)$. Therefore, the size of $D_n$ is bounded by $2^{O(n^c)}$. Now, $C_n$ has an advantage of at least $2^{-O(n^c)}$ in distinguishing between $f_i$ and $f_{i+1}$, whence it follows that $H(G_k)$ is bounded by $2^{O(n^c)} = 2^{O(k^\varepsilon)}$, a contradiction. □

## 3.3 The Main Theorem

In this section, we provide a self-contained proof of our main theorem. We will show that if there is a pseudorandom generator of hardness of the form $2^{n^\gamma}$, then for any $\Delta(\mathrm{EXP})$-martingale, we can construct a language in P/poly $\cap$ EXP on which the martingale does not succeed.

**Theorem 3.3** *If there is a pseudorandom generator $G=\{G_k : \{0,1\}^k \to \{0,1\}^{2k}\}$ of hardness $2^{k^\gamma}$ for some $\gamma > 0$ that is computable by a family of circuits of size $k^{O(1)}$, then* $\mu(\mathrm{P/poly}|\mathrm{EXP}) \neq 0$.

Before we proceed to the technicalities of the proof, let us spend some time trying to gain intuition about how the existence of a strong pseudorandom generator implies that P/poly does not have measure zero in EXP. We begin with the our first lemma, which follows easily from Markov's inequality.

**Lemma 3.4** *Let $\kappa$ be a martingale. For any string $u$ and any $\ell \in \mathbb{N}$, $r \in \mathbb{R}$,*

$$\left\| \{v \in \{0,1\}^\ell : \kappa(uv) \leq \left(1+\frac{1}{r}\right)\kappa(u)\} \right\| \geq 2^\ell \left(\frac{1}{r+1}\right).$$

In other words, Lemma 3.4 states that if we fix some string $u$ of length $2^n - 1$, and randomly choose a string $v$ of length $2^n$, then with probability at least $O(1/n^2)$, we will find a $v$ such that $\kappa(uv)$ is no more than $(1+1/n^2)\kappa(u)$. If we repeat this for each $n$, then we can find a sequence of $\{v_n\}$ such that the martingale $\kappa$ does not increase by more than a factor of $1+1/n^2$ at each $n$, along the sequence obtained by concatenating the $v_n$'s. Finally, we can define a language $L$ by letting $L^{=n} = v_n$, and it would follow that the martingale does not succeed on $L$, since the product $\prod \left(1+1/n^2\right)$ converges.

Of course, we do not obtain any upper bound on the complexity of the language thus constructed, since the $2^n$-bit strings $v_n$ were chosen completely at random. However, so far we have not used our hypothesis; we merely argued that if we keep selecting strings $v_n$'s at random, at every stage, we can find a reasonably "thick" fraction of strings that help us create some language $L$ on which the martingale does not succeed. Our hypothesis tells us that there is a very strong pseudorandom generator; can we *deterministically* choose the $v_n$'s from the range of the generator? This is precisely what we will do. First we note that if the pseudorandom generator does not work at any stage, then we can use the martingale to break the generator. Moreover, if we create a language $L$ this way, the seeds that produce a good sequence of $v_n$'s can be used as a succint representation of the language $L$ thus constructed. How strong do we want our generator to be? Our goal is to place the language $L$ in P/poly, so we would like the size of the seeds to be polynomial in $n$—i.e. we want a generator $G = \{G_n\}$ that stretches $n^{O(1)}$ bits into $2^n$ bits such that for any fixed any seed $s$, there is a circuit of size $n^{O(1)}$ that, on input $y \in \{0,1\}^n$, produces the $y$-th bit of $G_n(s)$. Moreover, we would like the generator to be secure against exponential-sized adversaries, since our martingale is an exponential-time algorithm. The construction of Proposition 3.2 provides precisely that, assuming the generator $G = \{G_k\}$ has hardness of the form $2^{k^\gamma}$ for some $\gamma > 0$. The proof that follows makes these arguments precise, and uses the notion of a *statistical test*, which is a circuit that attempts to break a pseudorandom generator.

**Proof.** (of Theorem 3.3) We begin by defining the following statistical test.

```
Algorithm $T_{\kappa,u}$, where $|u| = 2^n - 1$:
Input $v \in \{0,1\}^{2^n}$
Accept if and only if $\kappa(uv) \leq (1+1/n^2)\kappa(u)$.
```

Figure 3.1: Statistical Test $T_{\kappa,u}$

Let us now analyze the complexity of implementing the statistical test $T_{\kappa,u}$. Let $2^{\log^c |w|}$ bound the running time of a Turing machine that computes $\kappa(w)$. Since $|u| = 2^n - 1$ and $|v| = 2^n$, the statistical $T_{\kappa,u}$ can be computed in time $2^{O(\log^c 2^n)} = 2^{O(n^c)}$, and hence by a circuit of size $2^{O(n^c)}$. Furthermore, by Lemma 3.4, for any fixed $u$ of length $2^n - 1$, the probability, when $v$ is chosen uniformly at random from all strings of length $2^n$, that $T_{\kappa,u}$ accepts is at least $1/n^2$.

Let $G = \{G_k : \{0,1\}^k \to \{0,1\}^{2k}\}$ be the generator from the hypothesis of the theorem that is secure against circuits of size $2^{k^\gamma}$. By Proposition 3.2, this implies the existence of a pseudorandom generator $\widetilde{G} = \{\widetilde{G}_n : \{0,1\}^\ell \to \{0,1\}^{2^n}\}$ of hardness at least $2^\ell$, where $\ell = \Theta(n^c)$. Thus for any $u$ of length $2^n - 1$, if we choose a $v$ at random from the range of $\widetilde{G}_n$, then the probability that $T_{\kappa,u}$ accepts $v$ is at least $(1/n^2) - (1/2^{n^c})$, which is essentially $1/n^2$. It is clear, therefore, that the following algorithm produces a language $L$ on which $\kappa$ does not succeed. Finally, we will show that

```
Let $n_0$ be the first integer such that $N_0 = 2^{n_0}$ is large enough
    to apply the family $\widetilde{G}_n$ of generators.
Arbitrarily fix the memberships in $L$ of all strings of length less than $n_0$.
Set $u \leftarrow L^{<n_0}$
Set $n \leftarrow n_0$
while TRUE do:
    Let $v \in \{0,1\}^{2^n}$ be the first string in the range of $\widetilde{G}_n$ that $T_{\kappa,u}$ accepts.
    Set $L^{=n} \leftarrow v$.
    Set $u \leftarrow u \circ v$ (here $\circ$ denotes concatenation).
    Set $n \leftarrow n+1$.
```

Figure 3.2: Algorithm to produce a language that defeats a given martingale $\kappa$

the language $L$ thus defined is in P/poly $\cap$ EXP. Since the hypothesis of the theorem says that $G$ is computable by a family of polynomial-size circuits, Proposition 3.2 guarantees that there is a family of circuits $\{C_n\}$ of size $n^{O(c/\gamma)}$ such that for every $n$ and $x \in \{0,1\}^\ell$ and every $y \in \{0,1\}^{2^n}$, $C_n(x,y)$ outputs the $y$-th bit of $\widetilde{G}_n(x)$. In other words, if we hardwire the seed that produces the $v$'s chosen in Algorithm 3.2 into the family $\{C_n\}$ (at each $n$), we obtain a family of polynomial-size circuits that accepts $L$, i.e., $L \in$ P/poly. Furthermore, since the search for the $v$'s in Algorithm 3.2 is performed over the seed space of $\widetilde{G}_n$, which is of size $2^{O(n^c)}$, it also follows that $L$ may be accepted by a deterministic Turing machine running in time $2^{O(n^c)}$, i.e., $L \in$ EXP. Thus we have constructed a language $L \in$ P/poly $\cap$ EXP on which $\kappa$ does not succeed, whence it follows that $\mu(\text{P/poly}|\text{EXP}) \neq 0$. □

From the known equivalence of strong pseudorandom generators and strong one-way functions (see [ILL89, Hås90, HILL91]), we also have:

**Corollary 3.5** *If for some* $\gamma > 0$ *there exists a one-way function of security* $2^{n^\gamma}$*, then* P/poly *does not have measure zero in* EXP. □

Based on assumptions about the hardness of the *subset-sum* problem, Impagliazzo and Naor [IN89] show how to construct a pseudorandom generator in NC$^1$ (see also [NR95, BCK96]). Next we consider the hypothesis that there is a pseudorandom generator $G$ of exponential hardness that is computable efficiently in parallel, say in NC$^1$. Here, of course, *computability* refers to allowing efficient *indexing* in the sense of what Proposition 3.2 guarantees, i.e., suppose that there is a family of polynomial-size logarithmic depth circuits $\{C_n\}$ that, for any seed $x$ of length $n^{O(1)}$ and any $y \in \{0,1\}^n$, produces the $y$-th bit of the $2^n$-bit string $G_n(x)$.

**Theorem 3.6** *If there is a pseudorandom function generator of exponential hardness in* NC$^1$*, then nonuniform* NC$^1$ *is not measurable in* EXP.

## 3.4 Non-measurability of P/poly

In this section, we strengthen the conclusion of our main result from "not measure zero" to "not measurable at all." The key idea used is the fact that the operation of symmetric difference of languages acts like an "affine translation" that preserves measure.

**Lemma 3.7** *Let $\mathcal{C}$ be a proper subclass of* EXP *that is closed under symmetric difference. Then $\mathcal{C}$ does not have measure one in* EXP.

**Proof.** Suppose $\mathcal{C}$ does have measure one in EXP, that is, $\mathrm{EXP}-\mathcal{C}$ has measure zero in EXP. Let $\kappa$ be a martingale that succeeds on $\mathrm{EXP}-\mathcal{C}$.

Let $A \in \mathrm{EXP}-\mathcal{C}$. Define $\mathcal{C}_A = \{L \triangle A \mid L \in \mathcal{C}\}$. We claim that if $\mathrm{EXP}-\mathcal{C}$ has measure zero, then so does $\mathcal{C}_A$. To see this, first note that $\mathcal{C}_A \subseteq \mathrm{EXP}-\mathcal{C}$ since $\mathcal{C}$ and EXP are closed under symmetric difference. Now define a $\Delta(\mathrm{EXP})$ martingale $\kappa'$ that, on input $w$, outputs $\kappa(w \oplus v)$, where $v$ is the prefix of $\chi_A$ of length $|w|$, and where $\oplus$ denotes bitwise exclusive-or. Since $A \in \mathrm{EXP}$, it is easy for $\kappa'$ to compute $\chi_A$. Finally observe that for every $L \in \mathcal{C}$, $L \triangle A$ belongs to $\mathcal{C}_A$, so $\kappa$ succeeds on $L \triangle A$, and thus $\kappa'$ succeeds on $L$. ☐

Since the existence of a strong pseudorandom generator (of hardness at least polynomial) implies $\mathrm{EXP} \not\subseteq \mathrm{P/poly}$, Lemma 3.7 implies:

**Theorem 3.8** *If there exists a pseudorandom generator of hardness $2^{n^{\gamma}}$, for some constant $\gamma > 0$, then* P/poly *is not measurable in* EXP.

In fact, the proof of Lemma 3.7 can be extended to the case of subclasses of EXP that are closed under finite union and intersection:

**Lemma 3.9** *Let $\mathcal{C}$ be a proper subclass of* EXP *that is closed under finite union and intersection. Then $\mathcal{C}$ does not have measure one in* EXP.

**Proof.** If $\mathcal{C}$ has measure one in EXP, then so does co-$\mathcal{C}$, and hence $\mathcal{C} \cap$ co-$\mathcal{C}$. If $\mathcal{C}$ is closed under finite union and intersection, so is co-$\mathcal{C}$. Therefore, $\mathcal{C} \cap$ co-$\mathcal{C}$ is closed under symmetric difference, and Lemma 3.7 does the rest. ☐

**Corollary 3.10** *Let $\mathcal{C}$ denote any of* NP, coNP, $\Sigma_k^p$, $\Pi_k^p$, P/poly, *nonuniform* NC, BPP, PP, *or* PSPACE. *Then* $\mu(\mathcal{C}|\mathrm{EXP}) = 1 \iff \mathcal{C} = \mathrm{EXP} \iff \mathcal{C} \cap \textit{co-}\mathcal{C} = \mathrm{EXP}$. *In particular,* NP *has measure one in* EXP *iff* NP = EXP.

The last corollary answers an old open question in resource-bounded measure theory.

## 3.5 Natural Proofs

The technical concept at the heart of the paper by Razborov and Rudich [RR94] is the following. Define a *combinatorial property* to be a sequence $\Pi = [\Pi_n]_{n=0}^{\infty}$, where each $\Pi_n$ is a subset of the set $F_n$ of all $n$-variable Boolean functions. A language $A$ is *drawn from* $\Pi$ if for all $n$, the Boolean function given by $A^{=n}$ belongs to $\Pi_n$. The property $\Pi$ *is useful against* a class $\mathcal{C}$, or *diagonalizes against* of languages if no language drawn from $\Pi$ belongs to $\mathcal{C}$. When $\mathcal{C}$ is closed under finite variations, this is equivalent to diagonalizing *i.o.* against $\mathcal{C}$:

$$(\forall B \in \mathcal{C})(\exists^{\infty} n)\ B^{=n} \notin \Pi_n. \tag{3.2}$$

We remark that all of the natural properties constructed in [RR94] satisfy the stronger condition

$$(\forall B \in \mathcal{C})(\forall^{\infty} n)\ B^{=n} \notin \Pi_n. \tag{3.3}$$

We call this *diagonalizing a.e. against* $\mathcal{C}$.

The complexity of $\Pi$ is the complexity of the decision problem: given a Boolean function $f_n \in F_n$, is $f_n \in \Pi_n$? Finally, define the *density* of $\Pi_n$ by $\rho(\Pi_n) = \frac{\|\Pi_n\|}{2^N}$. The property is *large* if there exists a polynomial $p$ such that for all but finitely many $n$,

$$\rho(\Pi_n) \geq \frac{1}{p(N)}. \tag{3.4}$$

Put another way, the Boolean functions in $\Pi_n$ have *non-negligible* density in the space of all Boolean functions.

**Definition 3.4 (Natural Proof (cf. [RR94]))** *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes of languages. A combinatorial property $\Pi$ is $\mathcal{D}$-**natural against** $\mathcal{C}$ if $\Pi$ is large, belongs to $\mathcal{D}$, and is useful against $\mathcal{C}$.*

Razborov and Rudich show that several important separation results in complexity theory use techniques that construct natural properties. Their main theorem points out limitations of such techniques. The following improvement of their theorem from polynomial to quasipolynomial size bounds for $\mathcal{D}$ was noted by Razborov [Raz94]:

**Theorem 3.11** *If there exists a pseudorandom generator that is secure against circuits of size $2^{n^{\gamma}}$ for some $\gamma > 0$, then there is no large combinatorial property of quasipolynomial circuit complexity against* P/poly, *i.e. there is no* QP/poly-*natural proof against* P/poly.

### 3.5.1 The Proof of the Natural Proof Theorem

We briefly sketch a proof of the main result of Razborov and Rudich [RR94], setting the parameters for the slightly stronger version Theorem 3.11. The proof consists only of trivial modification of their parameters. We sketch the proof here for the sake of giving the reader a complete picture of how a natural proof can be used to break a pseudorandom generator, and for the sake of the proof of Theorem 3.16, which refers to certain aspects of this proof. We begin by noting the following, which is implicit in [RR94].

**Lemma 3.12** *If a natural property $\Pi$ (of arbitrary complexity) diagonalizes over* P/poly*, then for every polynomial $q$, there exist infinitely many $n$ such that for every circuit $C_n$ of size at most $q(n)$, $L(C_n)^{=n}$, treated as a $2^n$-bit string, does not belong to $\Pi_n$.*

**Proof.** Suppose to the contrary that for some polynomial $q$ there exists $n_0 \geq 0$ such that for all $n \geq n_0$, there exists a circuit $C_n$ of size $q(n)$ such that $L(C_n)^{=n} \in \Pi_n$. Define a language $L$ by letting $L^{=n} = L(C_n^*)^{=n}$ for all $n \geq n_0$, where $C_n^*$ denotes the lexicographically first circuit of size $p(n)$ that satisfies $L(C_n^*)^{=n} \in \Pi_n$. Clearly $L \in$ P/poly, yet $\Pi$ does not diagonalize over $L$, a contradiction. □

**Proof.** (of Theorem 3.11) Let the pseudorandom generator $G$ and $\varepsilon < \gamma$ be given. The goal is to show that for infinitely many $k$, $H(G_k) \leq 2^{k^\varepsilon}$. Let the natural property $\Pi$ against P/poly be such that each $\Pi_n$ has density $1/2^{(\log N)^c}$ and circuit size $2^{(\log N)^c} = 2^{n^c}$. For any $n$, set $k = n^{c/\varepsilon}$. By Proposition 3.2, using $G$ one can build a generataor $\widetilde{G}_n$ that stretches $k$ bits into $2^n$ bits, and such that for any fixed seed $x \in \{0,1\}^k$, there is a circuit of size $n^{O(c/\varepsilon)}$ that computes the $y$-th bit of $\widetilde{G}_n(x)$ for all $y \in \{0,1\}^n$. The $2^n$ bit string can be thought of as a segment of the characteristic sequence of a language that can be computed by a polynomial-size circuit. Thus every infinite sequence of seeds $\vec{x} = x_1, x_2, \ldots$ gives a language $L_{\vec{x}}$ that has a circuit family of a fixed polynomial size, say $q(n)$.

Now by Lemma 3.12, there are infinitely many $n$ such that for every seed $x$, $\widetilde{G}_n(x) \notin \Pi_n$. On the other hand, by the largeness of $\Pi$, it follows that a randomly chosen $f \in \{0,1\}^{2^n}$ belongs to $\Pi_n$ with probability at least $1/2^{O(n^c)}$. This shows that a circuit for $\Pi_n$ is a statistical test of size $2^{O(n^c)} = 2^{O(k^\varepsilon)}$ that distinguishes $\widetilde{G}_n(x)$ from a truly random Boolean function $f$, contradicting the hardness of the generator $G$. □

### 3.5.2 Martingales Yield Natural Proofs

We show that, in fact, Theorem 3.3 could have been proved by constructing a natural property that is useful against P/poly from a $\Delta(\mathrm{EXP})$-martingale that succeeds on P/poly$\cap$EXP. Thus the connection between measure theory and the theory of natural proofs is not merely informal, but very precise. In fact, in Section 3.7, we strengthen this connection by proving a weak converse of this result.

The essence of the construction of a natural property from a martingale is given by our key technical lemma, which has the idea that given a martingale $\kappa$ that succeeds on P/poly, we can build a combinatorial property that captures those Boolean functions on $\{0,1\}^n$ along which $\kappa$ makes too little income to succeed. This property then diagonalizes i.o. against the success class of the martingale, which contains P/poly. Since $\prod_n (1+1/n^2)$ converges, we can say that a return on capital of $1/n^2$, let alone losing money along a branch, is "too little income" for $\kappa$. Lemma 3.4 will guarantee that the density of these poor branches is at least $1/n^2 = 1/(\log^2 N)$, a notably greater density than that called "large" in Equation (3.4).

**Lemma 3.13** *If a $\Delta(\mathrm{EXP})$ martingale $\kappa$ succeeds on* P/poly$\cap$EXP *then for every polynomial $q$, there exist infinitely many $n$ and circuits $C_i$ of size at most $q(i)$, for $0 \leq i < n$, such that for all circuits $C_n$ of size at most $q(n)$,*

$$\kappa(u_0 \ldots u_n) > \left(1 + \frac{1}{n^2}\right) \kappa(u_0 \ldots u_{n-1}),$$

*where $u_i$ is the $2^i$-bit binary "characteristic string" that indicates the membership in $L(C_i)$ of $\{0,1\}^i$.*

**Proof.** Suppose not. Then there is a polynomial $q$ and constant $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, for every sequence of circuits $C_i$ of size at most $q(i)$, for $0 \leq i < n$, there exists a circuit $C_n$ of size at most $q(n)$ such that $\kappa(u_0 \dots u_n) \leq \left(1+\frac{1}{n^2}\right)\kappa(u_0 \dots u_{n-1})$, where the $u_i$'s have the same meaning as in the statement of the lemma.

We will build a language $L$ as follows: for strings of length less than $n_0$, membership in $L$ will be an arbitrary but fixed sequence. Let $\alpha = \kappa(u_0 \dots u_{n_0-1})$. Clearly $\alpha < \infty$. For $n \geq n_0$, we define $L^{=n}$ inductively. Let $u_0, \dots, u_{n-1}$ be the result of the recursively applying the construction to obtain $L^{<n}$; that is, $u_i = L^{=i}$. By assumption, there exists a circuit $C_n$ of size at most $q(n)$ such that $\kappa(u_0 \dots u_n) \leq \left(1+\frac{1}{n^2}\right)\kappa(u_0 \dots u_{n-1})$. Set $u_n = L(C^*)^{=n}$, where $C^*$ is the lexicographically first $C_n$ that satisfies this inequality (under some fixed encoding of circuits of size at most $q(n)$).

Clearly $L \in$ P/poly, since it can be accepted by the circuit family $[C_n]_{n=0}^{\infty}$. That $L \in$ EXP is immediate from the fact that finding the lexicographically first $C_n$ takes time at most $2^{q(n)+p(n)}$, where the running time to compute the martingale $\kappa$ determines $p(n)$. Finally,

$$\lim_{n\to\infty} \kappa(L^{\leq n}) \leq \alpha \prod (1+1/n^2) < \infty,$$

so $\kappa$ does not succeed on $L$, a contradiction. □

The remaining technical problem is to weave together the constructions in Lemma 3.13 for all polynomial bounds $q$. We do not know of a *uniform* way to choose the circuits $C_0, C_1, \dots, C_{n-1}$ promised by Lemma 3.13 over all $q$ and the infinitely-many $n$ for each $q$.

**Theorem 3.14** *If $\mu(\text{P/poly}|\text{EXP}) = 0$, then there is a* QP/poly-*natural property against* P/poly.

**Proof.** For each $k$, let $T_k$ be the infinite set of numbers $n$ promised by Lemma 3.13 for the bound $q(n) = n^k$. Set $T := \cup_k T_k$. For all $n \in T$, take the largest number $k \leq n$ such that $n \in T_k$, take the lexicographically first $C_0, \dots, C_{n-1}$ that works in Lemma 3.13, and define $U_{n-1}$ to be the concatenation of the corresponding $u_0, \dots, u_{n-1}$. For $n \notin T$, make some arbitrary choice such as $U_{n-1} = 0^{2^n-1}$. Finally, for all $n$ define

$$\Pi_n := \left\{ f_n : \kappa(U_{n-1} f_n) \leq \left(1+\frac{1}{n^2}\right)\kappa(U_{n-1}) \right\}.$$

Now, by Lemma 3.4, the property $\Pi = \{\Pi_n\}$ is large; in fact, it has density $1/\operatorname{poly}(n)$, not just $1/\operatorname{poly}(2^n)$. By the computability of the martingale $\kappa$, $\Pi_n$ can be recognized in quasi-polynomial time in $2^n$, given the $U_{n-1}$'s as advice. Equivalently, there is a family of circuits of size quasi-polynomial in $2^n$ that recognizes $\Pi_n$. Let $L$ be an arbitrary language in P/poly, and let $n^k$ be a bound on the size of a family of circuits to recognize $L$. Clearly, for all $n \in T_k$, $L^{=n} \notin \Pi_n$. Therefore, property $\Pi$ diagonalizes i.o. over P/poly. □

## 3.6 The Uniform Case and Honest Martingales

The next interesting question is whether Theorem 3.3 can be made to work under the hypothesis that for some $\gamma > 0$ there is a pseudorandom generator of hardness $2^{n^\gamma}$ *against uniform adversaries*, i.e. Turing machines as opposed to circuits. The main problem is that the natural property we construct in Theorem 3.14 is nonuniform, and this nonuniformity carries over to the statistical test constructed in the theorem of Razborov and Rudich, drawing on [GGM86]. That is, the property belongs to QP/poly. While the martingale is a uniform algorithm, why should the natural property derived from the martingale be nonuniform? We have not been able to obtain a QP-natural property under the hypothesis $\mu(\mathrm{P/poly}|\mathrm{EXP}) = 0$—the sticking point is that there seems to be no uniform description of the characteristic prefixes $u_0, \ldots, u_{n-1}$ obtained in applications of Lemma 3.13 to build the $\Pi_k$ that are interleaved in the proof of Theorem 3.14.

Interest in this problem led us to define the following "prefix-invariance" restriction on martingales, which also comes up naturally in the next section. We begin by formalizing the associated concept of a *betting strategy*.

**Definition 3.5 (Betting Strategy)** *A* ***betting strategy*** *is any function $b(\cdot)$ from $\{0,1\}^*$ to the closed interval $[-1 \ldots +1]$. The* ***martingale*** *$\kappa_b$* ***derived from*** *b is defined by $\kappa_b(\lambda) = 1$, and for all $w \in \{0,1\}^*$, $\kappa_b(w1) = \kappa_b(w)(1+b(w))$, $\kappa_b(w0) = \kappa_b(w)(1-b(w))$.*

For all $w$, let $x_w$ stand for the string indexed by the bit $c$ in $wc$, and let $n_w$ be the length of $x_w$; i.e., $n_w = \lfloor \log_2(|w|+1) \rfloor$. Intuitively, $b(w)$ is the signed proportion of current capital bet on the event that $x_w$ belongs to a given language $L$. A negative value of $b(w)$ indicates a bet that $x_w \notin L$. As remarked in Section 3.1.1, given a martingale $\kappa$, one can regard the function $b(w) \doteq (\kappa(w1) - \kappa(w))/\kappa(w)$ as the associated betting strategy. Henceforth we take "betting strategy" as the fundamental concept, and "martingale" as the derived one.

**Definition 3.6 (Honest Martingale)** *A martingale $\kappa : \{0,1\}^* \to \mathbb{R}$ is* ***honest*** *if it is derived from a betting strategy $b : \{0,1\}^* \to \mathbb{R}$, such that for all $w \in \{0,1\}^*$, the computation of $b(w)$ depends only on those parts of w that index strings of length $n_w$.*

With a few exceptions, most of the martingales implicitly constructed by Lutz et al. are honest. For honest martingales we note the following stronger form of Lemma 3.13:

**Lemma 3.15** *If an honest $\Delta(\mathrm{EXP})$ martingale $\kappa$ succeeds on $\mathrm{P/poly} \cap \mathrm{EXP}$ then for every polynomial q, there exist infinitely many n such that for all circuits $C_n$ of size at most $q(n)$, and all characteristic prefix strings $w \in \{0,1\}^{2^n-1}$, $\kappa(wu_n) \geq \left(1+\frac{1}{n^2}\right)\kappa(w)$, where $u_n$ is the binary characteristic string of length $2^n$ that represents the strings accepted and rejected by $C_n$.*

**Theorem 3.16** *If there is a pseudorandom generator $G = \{G_k\}$ of hardness $2^{k^\gamma}$ for some $\gamma > 0$ against uniform adversaries, then no honest $\Delta(\mathrm{EXP})$-martingale succeeds on $\mathrm{P/poly} \cap \mathrm{EXP}$.*

**Proof Sketch.** Given a honest $\Delta(\mathrm{EXP})$-computable martingale $\kappa$, for all $n$, let $w_n$ be some characteristic prefix of length $N-1$ such that $\kappa(w_n) > 0$. For all $n$, define

$$\Pi_n = \left\{ u \in \{0,1\}^N : \kappa(w_n u) \leq \left(1 + \frac{1}{n^2}\right) \kappa(w_n) \right\}.$$

The corresponding property $\Pi = \{\Pi_n\}$ is large and belongs to QP. By Lemma 3.15, and with the step of fixing "$u_0, \ldots, u_{n-1}$" in the proof of Theorem 3.14 now rendered unnecessary, it follows that $\Pi$ diagonalizes i.o. over P/poly. From the details of the Razborov-Rudich proof we sketched in Section 3.5.1, it can be verified that the statistical test constructed from $\Pi$ is computable by a probabilistic Turing machine in time less than $2^{k^\gamma}$. □

Theorem 3.16 strengthens Theorem 3.3 as well as the main theorem of Razborov and Rudich: if there is a pseudorandom generator of hardness $2^{n^\gamma}$ against uniform adversaries, then there is no P-natural or even QP-natural proof against P/poly $\cap$ EXP, not against all of P/poly. This leads to a sensitive and interesting point about the interplay between uniformity and nonuniformity. A QP/poly *martingale* is a martingale computed by circuits of quasipolynomial size; we also consider nonuniform martingales in the next section. The QP and QP/poly bounds, and the hardness bound, are tacit below:

(1) A nonuniform martingale that succeeds on P/poly yields a nonuniform natural proof against P/poly.

(2) A uniform martingale that succeeds on P/poly $\cap$ EXP also yields a nonuniform natural proof against P/poly.

(3) An honest uniform martingale that succeeds on P/poly $\cap$ EXP yields a uniform natural proof against P/poly.

(4) A uniform natural proof against P/poly $\cap$ EXP suffices to disprove the existence of pseudorandom generators secure against uniform adversaries.

(5) A nonuniform natural proof against P/poly $\cap$ EXP does nothing, because one exists—even diagonalizing a.e. against all r.e. sets. Given an enumeration $[M_i]$ of TMs, define for all $n$,

$$\Pi_n = \{w \in F_n : (\forall i \leq n) w \neq L(M_i)^{=n}\},$$

Then $\Pi \in$ P/poly because for strings of length $n$, i.e. for $w$ of length $N = 2^n$, we can "hardwire" the $n$-many characteristic sequences of how machines $Q_1, \ldots, Q_n$ behave at length $n$. Also each $\Pi_n$ has density $1 - n/2^N$, which is huge.

The last point indicates that much care is needed when using the natural proofs theory to talk about separations from *uniform* classes, whereas the measure theory is already tailor-made for uniformity. We ask, however, whether the theories are equivalent in the nonuniform case; i.e., whether every nonuniform natural proof $\Pi$ yields a nonuniform martingale that covers the class that $\Pi$ diagonalizes against.

## 3.7 Are martingales and natural properties equivalent?

The underlying idea behind the concepts of martingales and natural properties is a strong form of diagonalization, and it is natural to ask whether they are equivalent. In this section, we prove a partial converse of our main theorem. Our results emphasize that two parameters in the definition of natural properties are important: the *density* $\rho(\Pi_n)$ of the property $\Pi$, that is, $\|\Pi_n\|/\|F_n\|$, and whether $\Pi$ diagonalizes i.o. or a.e. (see Equations 3.2 and 3.3).

Our results here seem to need somewhat stronger closure properties of the class $\mathcal{D}$. Say a class $\mathcal{D}$ is *nice* if it is closed under parallel evaluation of polynomially many functions in $\mathcal{D}$, under finite composition, and under the operation of finding "majority." Clearly P/poly is a nice circuit class. Recall $n = \log N$, and that density $1/2^{O(n)}$ equals "large" in [RR94].

**Theorem 3.17** *Let $\mathcal{D}$ be a nice nonuniform class, and let $\mathcal{C}$ be any class of languages. Then:*

*(a) If there is a natural property $\Pi \in \mathcal{D}$ of density $1/n$ that diagonalizes a.e. against $\mathcal{C}$, then there is a martingale computable in $\mathcal{D}$ that succeeds on $\mathcal{C}$.*

*(b) If there is a natural property $\Pi \in \mathcal{D}$ of density $(1-1/n^{1+\varepsilon})$ that diagonalizes i.o. against $\mathcal{C}$, then there is a $\mathcal{D}$-martingale that succeeds on $\mathcal{C}$.*

*(c) If $\mathcal{D}$ is uniform, then the martingale is computed by a probabilistic $\mathcal{D}$-machine with negligible bounded error.*

**Proof.** Suppose we have a $\mathcal{D}$-natural property $\Pi$ that diagonalizes a.e. over $\mathcal{C}$, and let $A = \{A_n\}$ denote the algorithm (family of circuits) that decides $\Pi$. For every $n$, consider the full binary tree $T_n$ of depth $N = 2^n$ that has $2^N$ leaves in one-to-one correspondence with the members of $F_n$. Let $\Upsilon_n = F_n - \Pi_n$, and when $n$ is fixed or understood, let $\sigma = \|\Upsilon_n\|/2^N$ denote the density of $\Upsilon_n$.

For each $n$, the property $\Pi_n \subseteq F_n$ identifies a large subset of the leaves that are "avoided" by languages in $\mathcal{C}$. By the a.e. diagonalization condition, this means that for every $L \in \mathcal{C}$, and all but finitely many $n$, $L$ goes through a branch in $\Upsilon_n$ at length $n$. This is the only property of $\mathcal{C}$ that is used in the proof; the martingale works only with the information about $\Pi_n$ versus $\Upsilon_n$. Given unit capital at the root of $T_n$, the martingale we construct will adopt the following simple strategy: try to make profit along the paths to all leaves in $\Upsilon_n$, avoiding the leaves in $\Pi_n$. By the restriction on information, we allow that there may be no way for the martingale to distinguish among the leaves in $\Upsilon_n$, so the best it can achieve is to amass a capital of $2^N/\|\Upsilon_n\| = 1/\sigma$ at every leaf in $\Upsilon_n$.

Suppose the martingale is at an interior node $v$ of $T_n$. Let $V_0 = \{w \in F_n \mid w \sqsupseteq v0\}$ and $V_1 = \{w \in F_n \mid w \sqsupseteq v1\}$ denote the set of leaves in the subtrees $v0$ and $v1$, respectively. Let $p_0(v) = \|V_0 \cap \Upsilon\|/\|V_0\|$, $p_1(v) = \|V_1 \cap \Upsilon\|/\|V_1\|$. If the martingale could calculate $p_0(v)$ and $p_1(v)$ *exactly*, then it could set $\kappa(v0) = 2\kappa(v)\left(\frac{p_0}{p_0+p_1}\right)$ and $\kappa(v1) = 2\kappa(v)\left(\frac{p_1}{p_0+p_1}\right)$. This would ensure that each leaf in $\Upsilon$ ends up with a capital of $1/\sigma$.

The problem is that a martingale that runs in time poly$(N)$ cannot compute the membership in $\Upsilon_n$ of all the $2^N$ leaves. However, by taking polynomially many random samples at each interior node, a *randomized* machine $M$ can (with high probability) *estimate* the values $p_0(v)$ and $p_1(v)$ to a high degree of accuracy. Then $M$ can use these estimates in lieu of the actual values, and still obey the condition (3.1) that defines a martingale. This strategy is continued so long as the subtree below $v$ has more than $N^2$ nodes; when the subtree has atmost $N^2$ nodes, an exhaustive examination of all leaves is done and most of the capital is diverted towards the leaves in $\Upsilon_n$, leaving a tiny portion for the leaves in $\Pi_n$. This tiny amount is donated to ensure that leaves $z \in \Pi_n$ do not go to zero, so that the martingale may eventually succeed on languages $L \in \mathcal{C}$ with $z \sqsubseteq \chi_L$. To simplify the description of $M$ and the calculations below, we assume that if $M$ discovers that small subtree with $N^2$ nodes has *no* leaves that belongs to $\Upsilon_n$, it chooses some leaf arbitrarily and directs profits toward it. This "wastage" does not matter much to the profits on leaves that actually do belong to $\Upsilon_n$.

Let $q_0(v)$ and $q_1(v)$ denote, respectively, the estimates of $p_0(v)$ and $p_1(v)$ that are obtained by sampling. Via standard Chernoff-bound methods, one can show that upon taking poly$(N)$-many samples (for a suitably large polynomial), with probability $1-\exp(-N)$, the estimates are within an additive term of $\delta = 1/\mathrm{poly}(N)$ of the true values. The martingale will then adopt the policy that overestimation (by upto $\delta$) is harmless, but underestimation is dangerous. More precisely, the martingale will pretend that $q_0(v)$ and $q_1(v)$ underestimate $p_0(v)$ and $p_1(v)$, and will therefore use $q_0(v)+\delta$ and $q_1(v)+\delta$ as safer approximations to the actual values. It follows that

$$\frac{\kappa(v0)}{\kappa(v)} = 2\frac{q_0(v)+\delta}{(q_0(v)+\delta)+(q_1(v)+\delta)},$$
$$\frac{\kappa(v1)}{\kappa(v)} = 2\frac{q_1(v)+\delta}{(q_0(v)+\delta)+(q_1(v)+\delta)},$$

and that $\kappa(v0)+\kappa(v1) = 2\kappa(v)$.

Let $m = \lceil 2^N/N^2 \rceil$, let $\tau_1, \tau_2, \ldots, \tau_m$ denote the subtrees of $T_n$ at height $2\log N$ that contain $N^2$ leaves each. For each $i$, let $u_i$ denote the root of $\tau_i$, and let $p_i$ denote the probability $\|\mathtt{leaves}(\tau_i)\cap\Upsilon\|/N^2$. Let $\rho_i$ denote the density $\|\mathtt{leaves}(\tau_i)\cap\Upsilon\|/\|\Upsilon\|$; it is easy to see that $\rho_i = \frac{p_i}{p_1+p_2+\ldots+p_m}$. The total value of $\kappa(\cdot)$ at height $2\log N$ is exactly $2^{N-2\log N} = m$, and the strategy works if for each $i$, $\kappa(u_i) = \Omega(\rho_i m)$. We show:

*Claim. For every $i$, $\kappa(u_i) \geq 0.99\rho_i m$ whp.*

For any node $u$, let $\pi(u)$ denote the parent of $u$. Wlog. let $i=1$, and focus on the first subtree $\tau_1$ with $N^2$ leaves. Recall that by the simplifying assumption made above, for all $i$, $p_i \geq 1/N^2$. The worst case for $\tau_1$ is the following: at every ancestor $v$ of $\tau_1$, the subtree of $v$ containing $\tau_1$ had an underestimated probability, and the other subtree of $v$ had an overestimated probability. To wit: at the first level, $p_1$ is underestimated to be $p_1-\delta$, and $p_2$ is overestimated to be $p_2+\delta$; at the second level, $\frac{1}{2}(p_1+p_2)$ is underestimated to be $\frac{1}{2}(p_1+p_2)-\delta$, and $\frac{1}{2}(p_3+p_4)$ is overestimated to be $\frac{1}{2}(p_3+p_4)+\delta$, and so on. When this happens,

$$\begin{aligned}\kappa(u_1) &\geq \frac{p_1-\delta+\delta}{(p_1-\delta+\delta)+(p_2+\delta+\delta)}\cdot 2\kappa(\pi(u_1)) \\ &= 2\frac{p_1}{p_1+p_2+2\delta}\cdot\kappa(\pi(u_1))\end{aligned}$$

Similarly,

$$\kappa(\pi(u_1)) \geq 2\frac{p_1+p_2}{p_1+p_2+p_3+p_4+4\delta}\cdot\kappa(\pi(\pi(u_1)))$$

Continuing in this fashion $\log m$ times, we have

$$\kappa(u_1) \geq m\prod_{\ell=1}^{\log m}\frac{\sum_{i=1}^{2^{\ell-1}} p_i}{\left(\sum_{i=1}^{2^\ell} p_i\right)+2^\ell\delta}$$

Multiplying and dividing the above by $(p_1 + \ldots + p_m)$, and regrouping the terms,

$$\begin{aligned}
\kappa(u_1) &\geq m\frac{p_1}{p_1+\ldots+p_m}\prod_{\ell=1}^{\log m}\frac{\sum_{i=1}^{2^\ell} p_i}{\left(\sum_{i=1}^{2^\ell} p_i\right)+2^\ell\delta} \\
&= m\rho_1\prod_{\ell=1}^{\log m}\left(1-\frac{2^\ell\delta}{\left(\sum_{i=1}^{2^\ell} p_i\right)+2^\ell\delta}\right)
\end{aligned}$$

Since $p_i \geq 1/N^2 = p$ for all $i$,

$$\begin{aligned}
\kappa(u_1) &\geq m\rho_1\prod_{\ell=1}^{\log m} 1-\frac{2^\ell\delta}{2^\ell p+2^\ell\delta} \\
&= m\rho_1\left(1-\frac{\delta}{p+\delta}\right)^{\log m} \\
&\geq m\rho_1\left(1-\frac{1}{N^2}\right)^{N} && \text{setting } \delta = 1/N^4 \\
&= m\rho_1 e^{-1/N} \\
&\geq 0.99 m\rho_1 && \text{for } N \geq 100.
\end{aligned}$$

*End of Proof of Claim.*

By standard arguments about converting high-probability algorithms into nonuniform algorithms, this can be shown to give a $\mathcal{D}$ martingale that succeeds on $\mathcal{C}$.

If the only information used by the martingale is the fact that for every $L$ in $\mathcal{C}$, $L^{=n} \in \{0,1\}^N - \Pi_n$ (i.o./a.e.), then the factor of $1/\sigma = 1/(1-\rho(\Pi_n))$ is the best possible in stage $n$. If $\Pi$ is a.e. diagonalizing, then a density of $\Omega(1/n) = \Omega(1/\log N)$ for $\Pi_n$ gives a factor of $\Omega(1+1/n)$ in stage $n$, which suffices for the martingale to succeed on $\mathcal{C}$.

If $\Pi$ is merely i.o. diagonalizing, then the above factor seems insufficient. By a modification of the Borel-Cantelli lemma as applied to martingales [Lut92a] (see also [RS95]), it can be shown that if $\sum_n(1-\rho(\Pi_n))$ converges, then a successful martingale of equivalent nonuniform complexity can be constructed. For example, an i.o.-natural property $\Pi$ of density $1-\frac{1}{n^{1+\epsilon}}$ for some $\epsilon > 0$ against $\mathcal{C}$ would give a nonuniform martingale that succeeds on $\mathcal{C}$. $\square$

This partial converse brings out the importance of the actual *density* of the natural proof, and whether the diagonalization is i.o. or a.e. These are somewhat submerged in [RR94], but we note that all six of their examples diagonalize a.e., and the first four have density at least constant or $1-o(1)$. Hence there are reasons to investigate the effect of different densities.

A stronger converse question is whether the *non*-existence of strong pseudorandom generators implies that P/poly *does* have measure zero in EXP. From the non-existence it follows that given any generator of pseudorandom functions on $\{0,1\}^n$, a relatively small statistical test $T$ can distinguish them from truly random Boolean functions. However, $T$ need not have the sharp "all-or-nothing" form of the statistical test given by a natural proof, and this lack also hampers efforts to apply our proof idea of Theorem 3.17. In any case, there can be no simple answer, because there are oracles relative to which EXP is contained in P/poly—these give no pseudorandom generators but also P/poly has measure *one* in EXP!

## 3.8 Concluding Remarks

Our work in this chapter was concerned with the measure of the nonuniform (circuit) complexity class P/poly. We established that if sufficiently strong pseudorandom generators exist, then P/poly is not measurable in EXP.

How reasonable is the hypothesis that for some $\gamma > 0$, there is a pseudorandom generator of hardness $2^{n^\gamma}$? This author does not wish to make any assertion one way or the other. We do not present our main theorem as evidence that P/poly is non-measurable. As we mentioned in the Introduction, one of the central goals of this dissertation is to study the applications of pseudorandom generators in structural complexity theory. To that end, we present our main theorem as a very interesting structural-complexity consequence of the existence of strong pseudorandom generators.

Complexity theory has all along benefitted from theorems of the form "If pigs can whistle, then horses can fly." While such theorems don't prove any absolute result, they offer invaluable insight into the structure of complexity classes, and lead to further progress. Some of the more prominent whistling pigs include the hypotheses "$\mathrm{P} \neq \mathrm{NP}$," "PH is infinite," "one-way functions exist," and more recently, the intriguing measure-theoretic hypothesis "$\mu(\mathrm{NP}|\mathrm{EXP}) \neq 0$." Hypotheses about the existence of pseudorandom generators of considerable hardness have been considered before and some exciting results have been shown. For example, Yao [Yao82] (see also [BH89], [All89]) has shown that if generators of hardness $2^{n^\epsilon}$ exist, then randomized polynomial-time algorithms can be simulated in deterministic quasipolynomial time. The equivalence of our hypothesis about strong generators and the existence of strong one-way functions makes it an even more robust complexity-theoretic hypothesis. Hypotheses about strong one-way functions are very common in complexity-based cryptography; a particular example of recent interest is its application to the construction of zero-knowledge and interactive proofs [GMR89, GMW91]. We feel that the hypothesis about the existence of strong pseudorandom generators deserves to be studied further; in particular, its possible connection to average-case complexity theory seems particularly intriguing.

An interesting corollary of our main theorem in this section connects two measure-theoretic questions simultaneously with pseudorandom generators. Lutz [Lut92b] has shown that if a subclass $\beta$NP of NP does not have measure zero in EXP, then strong one-way functions and pseudorandom generators (of hardness considerably more than $2^{n^\epsilon}$) exist. Together with our result, this implies that any answer to the question of the existence of strong pseudorandom generators implies an interesting measure-theoretic consequence: if they do exist, P/poly is non-measurable, and if they don't exist, then $\beta$NP has measure zero. Moreover, we now have the following direct implication: if $\beta$NP does not have measure zero, then P/poly is not measurable. Of course, both the hypothesis and the consequence of this statement are open to interpretation.

One of the original motivations for this research was to find a sufficient condition for Lutz's hypothesis $\neg\mu(\mathrm{NP}|\mathrm{EXP}) = 0$. We briefly analyze whether Theorem 3.3 can be made to work with NP in place of P/poly. Our proof works by taking a hard pseudorandom generator $G$ and a *given* $\Delta(\text{EXP})$-computable martingale $\kappa$, and constructing a language $L \in \text{P/poly} \cap \text{EXP}$ on which $\kappa$ does not succeed. The languages $L$ involved are defined by nonuniform sequences of seeds $x$ for the "iterated generator" from Proposition 3.2. Thus it appears that our technique works only in the nonuniform setting, and completely new techniques will be needed to answer the question for NP.

Finally, we would like to say a few words concerning the relationship between the notions of measure zero and natural proofs. Our main theorem shows that proving that $\mathcal{C}$ has measure zero in EXP is at least as strong as exhibiting a QP-natural proof against $\mathcal{C}$. Our partial converse suggests that perhaps measure zero is a strictly stronger notion than a natural proof. We argue that it is indeed so, and with very good reason. Lutz's notion of measure satisfies the condition of measure conservation, i.e. it satisfies the property that EXP itself does not have measure zero in EXP. Suppose someone proves that P/poly has measure zero in EXP; not only does it prove some form of lower bound against P/poly, it immediately gives us a language $L \in \text{EXP} - \text{P/poly}$. However, a QP-natural property against P/poly, if one exists, does not give any upper bound on the complexity of a language shown to be outside P/poly, since there is no analogue of measure conservation in the natural proof framework. For this reason, we suggest that: *The notion of measure zero is more natural than a natural proof.*

# Chapter 4

# The Measure of $\mathrm{AC}^0$ and Friends

In the preceding chapter, we established a connection between the existence of pseudorandom generators and non-measure-zero of the circuit complexity class P/poly. Our main theorem in that chapter asserts, under the hypothesis about the existence of strong pseudorandom generators, that there is no $\Delta(\mathrm{EXP})$-martingale that succeeds on all of P/poly$\cap$EXP. We had to be content with a conditional result since the question of the existence of pseudorandom generators secure against circuits of nearly exponential size is wide open. Indeed, this question appears to be a quite formidable, since *necessary* conditions for the existence of such generators include major separations of complexity classes and significant complexity lower bounds on the Boolean circuit model. However, computational complexity theory has met with considerable success in the study of circuits of *constant depth,* and indeed strong pseudorandom generators against such circuits are known. Can we turn these around to prove an unconditional non-measurability result? We address this question in this chapter and prove a pair of complementary theorems that are as tight as one can hope for.

Before we present our main results of this chapter, we need to make (fairly long) digressions to set up the measure-theoretic framework in which we prove our results, and to give a brief overview of the history of lower bounds against constant-depth circuits. In Section 4.1, we introduce the various notions of measure on the complexity class P that we will be referring to in the subsequent sections. The generalization of resource-bounded measure to the complexity class P is not simple, and exhibits quite a few technical subtleties. Owing to notorious technical difficulties, researchers have been unable to devise a notion of measure on P that fully transfers the spirit and the basic axioms of measure at EXP. We remark that this is still an evolving notion, and the various definitions of "measure zero in P" and of related notions are slightly more complicated, although very natural.

In Section 4.2, we sketch briefly the history of developments in proving lower bounds against constant-depth circuits. In Section 4.3, we present the details of what turns out to be an important technical tool in our proofs—the pseudorandom generator secure against constant-depth circuits. This generator is due to Nisan [Nis91], and it is based on the strong lower bounds against constant-depth circuits. Besides presenting the details of the construction used in Nisan's generator, we sketch proofs of several facts about the generator, and present an enhancement of Nisan's generator that endows it with certain error-correcting properties.

Following the above technical preliminaries and historical remarks, we prove our main results in the Sections 4.4, 4.5, and 4.6. In Section 4.4, we consider one of the notions of measure on

P. We prove that for any martingale of the kind used to define this notion of measure, we can construct a language $L$ on which the martingale does not succeed. Moreover, we show that $L \in \mathrm{AC}^0_4[\oplus]$, that is, $L$ is accepted by a family of polynomial-size, depth-4 circuits of unbounded fanin built using AND, OR, NOT, and PARITY gates. In Section 4.5, we consider a more liberal notion of measure on P (one that allows more measure zero classes), and show that all of AC$^0$, in fact, the class of all languages accepted by families of nearly exponential-size circuits of constant depth, has measure zero under this notion. As a consequence, we are able to show that all of $\mathrm{NTIME}[n^{1/10-o(1)}]$ has measure zero in P. Recall that one of the central hypotheses in resource-bounded measure theory is the assertion $\mu(\mathrm{NP}|\mathrm{EXP}) \neq 0$. Scaled down to the (quasi)polynomial-time setting, this says that $\mathrm{NTIME}[(\log n)^{O(1)}]$ does not have measure zero in quasipolynomial time, $\mathrm{QP} = \mathrm{DTIME}[2^{(\log n)^{O(1)}}]$. Our result is a refutation of this low-level analogue of the hypothesis $\mu(\mathrm{NP}|\mathrm{EXP}) \neq 0$; in fact, our result refutes an exponentially weaker hypothesis. In sharp contrast, we construct in Section 4.6, for any martingale used to define the same notion of measure, a language $L \in \mathrm{AC}^0_4[\oplus]$ on which the martingale does not succeed. A crucial technical ingredient in all three results of this chapter is the marvelous pseudorandom generator constructed by Nisan, building on the strong lower bounds against constant-depth circuits.

In Section 4.7, we conclude with some remarks on the role of pseudorandomness in the results of this chapter.

## 4.1 Measure on P: Preliminaries

To define a measure on P, one would naturally try to extend the ideas used to define measurs on E and EXP, by scaling down the complexity bounds. For example, we might try to define the notion of $\Delta(\mathrm{P})$-measure zero via martingales that can be computed in time $(\log|w|)^{O(1)}$ on the prefix $w$ (the node $w \in \Omega$, see Section 3.1.1). However, it turns out to be not so simple: under this definition, there seems to be no way to prove the *measure conservation theorem*, i.e., to show that P itself does not have measure zero.

Informally described, the way to prove measure conservation at EXP works as follows. Given a martingale $\kappa$, the goal is to construct a language in EXP on which $\kappa$ does not succeed. The crucial observation is that for any node $w \in \Omega$, at least one of $\kappa(w0)$ or $\kappa(w1)$ is bounded above by $\kappa(w)$, and precisely which one is can be identified in time $2^{(\log|w|)^{O(1)}}$, which is $2^{|x|^{O(1)}}$, where $x$ is string whose membership is encoded in the last bit of $wb$. The language $L$ will be defined by the following algorithm. Given input $x$, let $w$ denote the portion of the characteristic sequence of $L$ that gives the membership in $L$ of all strings that lexicographically precede $x$. Decide $x \in L \iff \kappa(w1) < \kappa(w)$. Of course, this entails the computation of $\kappa(w1)$ and $\kappa(w)$, for which we (presumably) need to know what $w$ is. When the computation of $\kappa(w)$ or $\kappa(w1)$ needs some bit of $w$, that gives the membership in $L$ of some $y \prec x$ (where $\prec$ denotes lexicographic order), we recursively apply the algorithm. It is clear that the algorithm can be implemented in time $2^{|x|^{O(1)}}$, where the constants are determined by the time required to compute the martingale (which, by hypothesis is of the form $2^{|x|^{O(1)}}$).

Let us try to adopt the same procedure for martingales that run in time $(\log|w|)^{O(1)}$ time, i.e., in time $|x|^{O(1)}$, where $|w|+1$ is the number whose binary representation is $1x$. Whenever the computation of $\kappa(w)$ requires some bit of $w$, we will attempt to recursively compute $\kappa$ on some prefix of

$w$. Suppose that, for any $z$, the computation of $\kappa(z)$ begins by requesting the second-from-last bit of $z$; i.e., if $z = z_0 z_1 \dots z_N$, then the bit $z_{N-1}$ will be requested. Then the length of the recursive chain of calls to compute $\kappa(w)$ will be $|w| \approx 2^{|x|}$, and cannot be done in time polynomial in $|x|$, which is needed to place $L$ in P.

We note that Allender and Strauss [AS94b] were the first to consider the idea of defining a measure on P. Allender and Strauss [AS94a] realized that measure conservation was the stumbling block, and found a way to circumvent the problem outlined above. Subsequent refinements and robustness theorems appear in [AS95, Str96]; we only summarize the notions relevant to our work. Some of our notations are also our inventions.

The solution to the measure conservation issue suggested by Allender and Strauss is simple, natural, and elegant. It involves the following notion of a small *dependency set*:

**Definition 4.1** *A martingale $\kappa$ is said to have* ***polylogarithmic size dependency set*** *if for every $N$ there exists a set $S_N \subseteq \{0,\dots,N-1\}$ of size $(\log N)^{O(1)}$ such that for every every $w$ of length $N$ and every prefix $z \sqsubseteq w$ of length $M \in S_N$, the computation of the value $\kappa(z)$ depends on only those bits of $z$ whose indices are listed in the set $S_N \cap \{0,\dots,M\}$.*

An equivalent viewpoint is the following: at any level $x$ of the tree $\Omega$, there is a *dependency set* $G_x \subseteq \{y \prec x\}$ that is computable in time polynomial in $|x|$ (in particular, $G_x$ has at most $|x|^{O(1)}$ elements), such that for all $y \in G_x$, $G_y \subset G_x$. For any $w$ s.t. $|w|+1 = 1x$, the capital $\kappa(w)$ is computable in time $|x|^{O(1)}$, and depends only on those bit positions of $w$ specified in $G_x$.

**Definition 4.2** *A martingale $\kappa$ that can be computed under the above restrictions (of running time and dependency set size $(\log|w|)^{O(1)}$ in the length of its input $w$) is called a $\Gamma_\kappa(\mathrm{P})$-**martingale**.*

An analogous definition arises if we let the martingale output its bet $\beta(w) \doteq \kappa(w1) - \kappa(w0)$ instead of the capital $\kappa(w)$ (see Section 3.1.1, page 40 for detailed explanations). For martingales that operate in time exponential in $|x|$, the two notions coincide; however, for martingales whose running times and dependency set sizes are restricted to be polylogarithmic, the two notions seem vastly different. We will comment more about this in Section 4.7.

**Definition 4.3** *A martingale $\kappa$ whose induced betting function $\beta \doteq \beta_\kappa$ can be computed under the above restrictions (of running time and dependency set size $(\log|w|)^{O(1)}$ in the length of its input $w$) is called a $\Gamma_\beta(\mathrm{P})$-**martingale**.*

In either case, the notion of *success* will be defined with respect to the capital function $\kappa$; this is clearly valid even if the computation of a martingale outputs only the bets, since the sequence of bets implicitly defines the sequence of the capital values at each node. When the difference between the two notions is irrelevant, such as in defining the notion of success, we simply write "$\Gamma(\mathrm{P})$-martingale" to denote either. Analogous to measure at exponential time, we say a $\Gamma(\mathrm{P})$-martingale $\kappa$ *succeeds on* a language $A$ if $\lim_{w \sqsubseteq \chi_A} \kappa(w) = +\infty$. Finally, we define two notions of measure zero at P by writing $\mu_{\Gamma_\kappa(\mathrm{P})}(\mathcal{C}) = 0$ (resp. $\mu_{\Gamma_\beta(\mathrm{P})}(\mathcal{C}) = 0$), if there is a $\Gamma_\kappa(\mathrm{P})$-martingale (resp. $\Gamma_\beta(\mathrm{P})$-martingale) that succeeds on $\mathcal{C}$.

As it turns out, that is not all. In their paper on the study of robustness of the notion of measure on P, Allender and Strauss raise yet another question, which we motivate as follows. Recall that a $\Gamma(\mathrm{P})$-martingale, on an input node $w$, gets to look only at a small subset of the input bits. In particular, this subset could be entirely disjoint from the set of input bits needed to compute the martingale at the nodes $w0$ and $w1$. How reasonable is it to require the martingale to satisfy the exact average law $\kappa(w) = (\kappa(w0)+\kappa(w1))/2$, given that the computation of $\kappa(w0)$ or $\kappa(w1)$ (or that of $\beta(w0)$ and $\beta(w1)$) has *no idea* of what $\kappa(w)$ is? What if we relax this condition to $\kappa(w) \geq (\kappa(w0)+\kappa(w1))/2$, allowing the martingale to "throw away" some of its money, going from $w$ to its children? This notion is called a *super-martingale* in classical measure theory literature. Again, for martingales of the $\Delta(\mathrm{E})$ or $\Delta(\mathrm{EXP})$ kind, this relaxation doesn't achieve anything new. However, given the restriction on dependency sets, it appears that this relaxation might be useful in proving more measure zero results, and this is indeed so. Allender and Strauss [AS95] show that one can construct martingales that take advantage of this relaxation, *if, in addition*, we redefine success as achieving a *lim sup* of $+\infty$ rather than achieving a limit of infinity. This is very natural, since such martingales very often operate in "stages," where the initial capital of \$1 is divided among the stages, and each stage starts with a very small capital (e.g., a capital of $O(1/n^2)$ for the $n$-th stage, so that the total capital used is \$1). At the point of transition from the $n$-th stage to the $(n+1)$-st, the martingale will throw away all its earnings from stage $n$, and start all over again with the puny $O(1/n^2)$ capital. As an example, they showed that the class SPARSE of all sparse languages (cf. Chapter 2) can be covered by a $\Gamma_\kappa(\mathrm{P})$-martingale that succeeds in the lim sup.

We will write $\mu^*_{\Gamma_\kappa(\mathrm{P})}(\mathcal{C}) = 0$ if there is a $\Gamma_\kappa$ martingale that succeeds on $\mathcal{C}$ in the lim sup sense, and we will write $\mu^*_{\Gamma_\beta(\mathrm{P})}(\mathcal{C}) = 0$ if there is a $\Gamma_\beta$ martingale that succeeds on $\mathcal{C}$ in the lim sup sense.

Clearly, $\mu_{\Gamma_\kappa(\mathrm{P})}(\mathcal{C}) = 0$ is stronger than $\mu^*_{\Gamma_\kappa(\mathrm{P})}(\mathcal{C}) = 0$, which is stronger than $\mu^*_{\Gamma_\beta(\mathrm{P})}(\mathcal{C}) = 0$. The relationship of $\mu_{\Gamma_\beta(\mathrm{P})}(\mathcal{C}) = 0$ to these is not clear; it is clearly stronger than $\mu^*_{\Gamma_\beta(\mathrm{P})}(\mathcal{C}) = 0$, and, by our result in Section 4.5, $\mu_{\Gamma_\beta(\mathrm{P})}(\mathcal{C}) = 0$ appears weaker than $\mu^*_{\Gamma_\kappa(\mathrm{P})}(\mathcal{C}) = 0$.

As a sneak preview into the rest of this chapter, we state our main results briefly. Recall that $\mathrm{AC}^0[\oplus]$ denotes the class of languages accepted by nonuniform families of constant depth, polynomial-size circuits of unbounded fanin that use AND, OR, NOT, and PARITY gates; recall also that $\mathrm{AC}^0_4[\oplus]$ denotes the restriction of $\mathrm{AC}^0[\oplus]$ to depth 4 circuit families. Also, let $\widehat{\mathrm{AC}^0_d}$ denote the class of languages accepted by nonuniform families of circuits with AND, OR, and NOT gates that have depth $d = d(n) = o(\log n/\log\log n)$ and size $2^{n^{\frac{1}{2d+6+\delta}}}$ for some $\delta > 0$. The class $\widehat{\mathrm{AC}^0_2}$ contains $\mathrm{NTIME}[n^{1/(10+\delta)}]$.

**Main Results of this Chapter:**

(1) $\mu^*_{\Gamma_\kappa(\mathrm{P})}(\mathrm{AC}^0_4[\oplus]) \neq 0$.

(2) For any $d = o(\log n/\log\log n)$, $\mu_{\Gamma_\beta(\mathrm{P})}(\widehat{\mathrm{AC}^0_d}) = 0$.

(3) $\mu_{\Gamma_\beta(\mathrm{P})}(\mathrm{AC}^0_4[\oplus]) \neq 0$.

## 4.2 Lower Bounds against Constant-Depth Circuits: A Brief History

Perhaps the biggest success story of complexity theory, after the seminal time- and space-hierarchy theorems of Hartmanis and Stearns [HS65], is the nearly optimal lower bounds we know today against constant-depth Boolean circuits. In 1981, Furst, Saxe, and Sipser [FSS81] (see also [FSS84]) initiated the line of research that focused on proving stronger and stronger lower bounds against constant-depth circuits. Their original motivation was to construct an oracle relative to which the polynomial-time hierarchy (PH) could be separated from PSPACE, and hopefully, relative to which the levels of the polynomial-time hierarchy could be proved to be distinct. They showed the fundamental connection between this problem and that of proving lower bounds against constant-depth circuits. They also proved that no circuit of size $n^{O(1)}$ and of some fixed depth can compute the parity function on $n$ bits. This was slightly weaker than what was required to separate PH from PSPACE. Nevertheless, their proof introduced, for the first time, the technique of *random restrictions* which was used in the subsequent strengthenings of their result as well as many others. In 1985, Yao [Yao85] significantly improved their result and proved a weakly exponential lower bound on the size of a constant depth circuit that computes parity. While still not optimal, Yao's bound was sufficient for the construction of an oracle relative to which the levels of PH are distinct, and PH $\neq$ PSPACE. In the following year, Cai [Cai86] (see also [Cai89]) and Håstad [Hås86] (see also [Hås89]) improved Yao's result in two slightly different directions: Håstad obtained the nearly optimal lower bound of $2^{\frac{1}{10}n^{1/(d-1)}}$ on the size of a depth $d$ circuit to compute the parity of $n$ bits. Cai proved the remarkably strong result that any circuit of depth $d$ and size of the form $2^{n^{\varepsilon}}$, where $\varepsilon = \varepsilon(d)$, can not only not compute parity, but also must err on asymptotically 50% of all inputs. Cai's result implied that not only is there an oracle relative to which PH $\neq$ PSPACE, but indeed PH $\neq$ PSPACE relative to almost every oracle! Cai's proof technique used the notion of a *decision tree*, which, as a computational model, was ideally suited to proving the asymptotically 50% error result for constant-depth nearly exponential-size circuits. The main technical tool in Håstad's strong lower bound proof is called the *Switching Lemma*, which has come to be recognized as a powerful technique in complexity theory. It is possible to combine the two techniques, and to prove a decision-tree version of the Switching Lemma with parameters essentially as strong as obtained by Håstad.

## 4.3 Nisan's Pseudorandom Generator

The relevance of lower bound results that show an asymptotically 50% error rate was not very well understood (except that it implied, in the case of constant-depth circuits, a random oracle separation of PH from PSPACE) until the breakthrough work of Nisan and Wigderson [Nis91, NW88, Nis92]. Nisan and Wigderson proved the remarkable connection that shows how to transform computational hardness into (pseudo)randomness. They showed how a lower bound that shows asymptotically 50% error (of the form proved by Cai [Cai86, Cai89]) can be used to build a pseudorandom generator secure against the class against which the lower bound is proved. Nisan [Nis91] derived such bounds for constant-depth circuits from Håstad's Switching Lemma, and constructed an extremely strong pseudorandom generator that is secure against constant-depth circuits of nearly exponential size.

It turns out that Nisan's pseudorandom generator plays a crucial role in all three results of this chapter. In the proofs of our results, we use several facts about the technicalities of this generator;

therefore we present all relevant details of Nisan's generator in this section. In addition, we prove some facts about Nisan's generator that are not very obvious, and present one technical enhancement of Nisan's generator. These facts are probably folklore, but we haven't been able to find their proofs in the literature; we present proofs that were discovered independently by us. The enhancement of Nisan's generator that we present in this section appears in [CSS96].

**Theorem 4.1 (Nisan[Nis91])** *Let $\varepsilon > 0$, and let $D \geq 2$ be an integer. For a sufficiently large integer $N$ and $\log^D N \leq M < L < N$, let $G$ be an $L \times N$ matrix ($L$ rows and $N$ columns) over $\mathbb{Z}_2$ such that* (1) *Every column of $G$ has at least $M$ 1's;* (2) *Any two distinct columns of $G$ have at most $\log N$ 1's in common. Then for any circuit $C$ of depth $D$ and size $2^{\frac{1}{10}M^{\frac{1}{D+1+\varepsilon}}}$ with $N$ inputs,*

$$\left| \Pr_{y\in\{0,1\}^N}[C(y)=1] - \Pr_{s\in\{0,1\}^L}[C(sG)=1] \right| \leq \frac{1}{2^{\frac{1}{10}M^{\frac{1}{D+1+\varepsilon}}}},$$

*where $sG$ denotes multiplication of the row vector $s$ by the matrix $G$ over $\mathbb{Z}_2$.*

The matrix $G$ used in Nisan's generator is constructed as follows. Choose $M$ to be a prime power, and let $L = M^2$. The rows of $G$ are indexed by pairs $(a,b) \in GF(M) \times GF(M)$, and the columns of $G$ are indexed by all polynomials in $GF(M)[x]$ of degree at most $\log N$. There are at least $M^{\log N}$ such polynomials, which is considerably more than $N$ since $M \geq \log^D N$. The entry $G[(a,b),p]$ is set to 1 if and only if $p(a) = b$. It is clear that every column has at least $M$ 1's. Moreover, any two distinct columns $p$ and $q$ can have no more than $\log N$ 1's in common, since the common 1's correspond to roots of $p-q$, of which there are at most $\log N$.

We are now ready to prove two facts about Nisan's generator; these facts will be important for the results of Section 4.4 and Section 4.6. The first fact we demonstrate states that the bits of the output of Nisan's generator possess some form of limited independence between them. We will first formally define the notion.

**Definition 4.4** *A collection of $n$ 0-1 random variables $X_1, X_2, \ldots, X_n$ are said to be $k$-**wise independent** (for $k \leq n$) if for every $k$ indices $1 \leq i_1, i_2, \ldots, i_k \leq n$ and $b_1, b_2, \ldots, b_k \in \{0,1\}$, $\Pr[\bigwedge_{j=1}^k X_{i_j} = b_j] = \prod_{j=1}^k \Pr[X_{i_j} = b_j]$.*

**Fact 4.2** *Let $Y_1, Y_2, \ldots, Y_N$ denote the 0-1 random variables that denote the output bits of Nisan's generator when a seed $s \in \{0,1\}^L$ is chosen uniformly at random. Then the $Y_i$'s are all identically distributed with $\Pr[Y_i = 0] = \Pr[Y_i = 1] = 1/2$ for all $i$, and moreover, the $Y_i$'s are $(M/\log N)$-wise independent.*

**Proof.** Since every column of $G$ is non-zero, it is clear that when $s$ is chosen uniformly at random from $\{0,1\}^L$, each $Y_i$ is equally likely to be a 0 or a 1, since $Y_i$ is just the inner product mod 2 of $s$ with the $i$-th column of $G$. Moreover, since each column has at least $M$ 1's, and since any two columns intersect in at most $\log N$ positions, the $M$ 1's present in any column cannot be "covered" by the sum of fewer than $M/\log N$ columns. It follows that any set of $(M/\log N)$ columns of $G$ are linearly independent. Now for any set of $K \leq (M/\log N)$ indices $i_1, i_2, \ldots, i_K$ and any value $b_1, b_2, \ldots, b_K \in \mathbb{Z}_2$, the probability that $Y_{i_1} = b_1, Y_{i_2} = b_2, \ldots, Y_{i_K} = b_K$ equals the probability that a randomly chosen vector $s \in \{0,1\}^L$ falls in the intersection of the $K$ affine hyperplanes defined by the equations $G_{1,i_j}s_1 + G_{2,i_j}s_2 + \ldots + G_{L,i_j}s_L = b_j$ for $j = 1,2,\ldots,K$. Since these equations are linearly independent, this probability is precisely $1/2^K$, which is the product of the probabilities $\Pr[Y_{i_j} = b_j]$. □

The next fact about Nisan's generator concerns its computability. It is clear that all the $N$ bits of the output may be computed by a circuit of size $O(N+L)$ that uses only parity gates. However, in some situations, we are interested in a different kind of computability. Consider the case where $N = 2^n$, $L = (\log N)^{O(1)} = n^{O(1)}$, and where the bits of the output are indexed by strings in $\{0,1\}^n$, and suppose that we have a fixed seed $s \in \{0,1\}^L$. Is there a constant-depth $n^{O(1)}$-size circuit that takes as input $y \in \{0,1\}^n$ and computes the $y$-th bit of $sG$? The next fact asserts that indeed there is such a circuit.

**Fact 4.3** *Let $n = \log N$. For any fixed seed $s \in \{0,1\}^L$, there is an $n$-input circuit $C_s$ of depth 4 and size $(n+L)^{O(1)}$ that uses only AND and PARITY gates and that, on input $y \in \{0,1\}^n$, produces the $y$-th bit of $sG$.*

**Proof.** The input $y$ to $C_s$ denotes the index of some column of $G$, and thus corresponds to a polynomial in $GF(M)[x]$. Wlog. we may assume that the bits of $y$ may be divided into blocks of $\lceil \log_2 M \rceil$ bits each, and that each block gives one coefficient of the polynomial $p_y$ that corresponds to the column $y$ of $G$. Notice that $M = \sqrt{L}$, so each coefficient is only $O(\log L)$ bits long. Let $k$ denote the number of coefficients. The first stage of $C_s$ computes in parallel $p_y(z)$ for every $z \in GF(M)$. For each $z$, $C_s$ has $k$ pre-computed formulas $\phi_{z,1},\ldots,\phi_{z,k}$ over the basis AND, PARITY, and **1**. The formula $\phi_{z,i}$ takes the $i$-th coefficient $p_{y,i}$ of $p_y$ and outputs $p_{y,i}z^i$. Clearly the size of each $\phi_{z,i}$ is only $O(M \log M)$. The $\phi_{z,i}$'s are arranged to be PARITIES-OF-AND's. In the next stage, the terms $p_{y,i}z^i$ of the evaluation $p_y(z)$ are added using PARITY gates. In the third stage, each $p_y(z)$ is fed into $M$ formulas $\psi_w$ that decide if $p_y(z) = w$. Again, each $\psi_w$ is a formula of size $O(M)$, and there are exactly $M^2 = L$ of these. The outputs of these formulas are precisely the bits of the $y$-th column of $G$. The $\psi_w$'s can be arranged to be AND'S-OF-PARITIES. Finally, precisely those bits of the $y$-th column of $G$ are chosen that are specified by the seed $s$ (in fact, we could have restricted our $\psi_w$'s to compute only those), and their parity sum taken using a PARITY gate with $L$ inputs. Thus the circuit has size $(n+L)^{O(1)}$, and has the following levels of gates: AND-PARITY-PARITY-PARITY-AND-PARITY, which can clearly be collapsed to AND-PARITY-AND-PARITY for a total depth of only four. □

### 4.3.1 Endowing Nisan's Generator with Error-Correcting Properties

A *linear code* of length $N$, dimension $L$ is a subspace $\mathcal{C}$ of $\mathbb{Z}_2^N$ of dimension $L$. The elements of a code are called *codewords*. $\mathcal{C}$ is said to have *minimum distance* $\Delta$ if any two nonzero codewords in $\mathcal{C}$ differ in at least $\Delta$ positions. If $\mathcal{C}$ is a linear code, then the minimum distance of $\mathcal{C}$ equals the minimum Hamming weight of any nonzero codeword in $\mathcal{C}$.

It is easy to see that the range of Nisan's generator is a linear code of length $N$ and dimension $L$. From Fact 4.2, it also follows that the code has an average Hamming weight of $N/2$. However, in general, the code doesn't necessarily have large distance, i.e. large minimum weight. It turns out that we need this additional property for our main result of Section 4.5.

We will show how to suitably redesign the matrix used in Nisan's generator to produce a linear code of minimum distance roughly $N/L$. Of course, this only makes sense if $N/L$ is reasonably large—for example, if $N/L$ is less than 3, there doesn't seem to be any conceivable use of this. In our application, $N/L = \Omega(\log N)$. Our modification will still satisfy the hypothesis of Theorem 4.1, so the output of the generator will look random to circuits of certain size and depth as guaranteed in Theorem 4.1.

Let $\Delta = \lfloor N/L \rfloor$. We will view each codeword in the range of the generator as the concatenation of $\Delta$ *blocks* of $L$ bits each. We will show how to construct the matrix $G$ used in Nisan's generator so that every nonzero codeword has at least one 1 in every block of $L$ bits. Thus, for every $k$, every pair of codewords in the range of Nisan's generator differ in at least $k$ of their first $kL$ bits. We follow the error-correction conventions for matrix multiplication: The seed/message is a row vector, and is multiplied on the right by a generator with more columns than rows, yielding a longer pseudorandom codeword. (This matrix is transposed compared with Nisan's paper [Nis91].)

Find a prime power $M \geq \log^D N$, and let $L = M^2 + 1$. Define an $(L-1) \times N$ matrix $G$ as follows. The rows of $G$ will be indexed by pairs $(a,b) \in GF(M) \times GF(M)$, and the columns of $G$ will be indexed by all polynomials in $GF(M)[x]$ of degree at most $\log N - 3$, of which there are at least $M^{\log N}$, which is considerably more than $N$. The entry $G[(a,b),p]$ is set to 1 if and only if $p(a) = b$. It is clear that every column has exactly $M$ 1's. Moreover, any two distinct columns $p$ and $q$ can have no more than $\log N - 3$ 1's in common, since the common 1's correspond to roots of $p - q$, of which there are at most $\log N - 3$. Now tack on a row of all 1's to the matrix; we get $M^2 + 1 = L$ rows, at least $(M+1)$ 1's per column, and at most $(\log N - 2)$ 1's in common between any pair of columns. Note that we've satisfied the requirements of Nisan's theorem with room to spare: We can add one 1 per column and have at most $\log N$ 1's in common between two columns, and we can remove one 1 per column and have at least $M$ 1's per column.

We now have an $L \times N$ matrix. Regard the matrix as the concatenation of $\Delta$-many $L \times L$ submatrices, with perhaps some leftover columns. The next step is to make each of these submatrices nonsingular, one submatrix at a time, by changing at most one bit per column. The Gaussian elimination argument of Figure 4.1 achieves this.

```
For each submatrix
    For $I = 2$ to $L$
        Find the unique linear combination of the first $I - 1$ columns that
            agrees with the $I$-th column in the first $I - 1$ entries
        If also the $I$-th entries agree, then flip the $I$-th bit of the $I$-th column
```

Figure 4.1: Algorithm to make the generator submatrices nonsingular

For each $L \times L$ submatrix of $G$, it is easy to see, by induction on $I$, that carrying out the algorithm of Figure 4.1 makes the submatrix nonsingular. Thus we get a matrix $G$ that includes the concatenation of $\Delta$-many nonsingular $L \times L$ matrices whose row space (which comprises bit strings of length $N$)

is secure against circuits of depth $D$ and size $2^{\frac{1}{10}m^{\frac{1}{D+1+\varepsilon}}}$. Moreover, for every integer $k$, the first $kL$ columns of every nonzero codeword in the row space of $G$ have Hamming weight $k$. This is because if $s$ is a nonzero seed vector of length $L$, then the codeword $sG$ equals the concatenation of the product of $s$ with each of the $k$ nonsingular $L \times L$ submatrices of $G$, and every such product is nonzero. It is also clear that this entire construction can be carried out in time polynomial in $N$.

We point out next that decoding this code is also easy. Given any word $X \in \{0,1\}^K$, for $K \leq N$, there is at most one codeword $Z$ that is within a distance of at most $\lfloor K/2L \rfloor$ from $x$ in its first $K$ positions (assume, for simplicity, that $K/L$ is odd, so that $\lfloor K/2L \rfloor = (K/L-1)/2$). To find $Z$, we divide $X$ into $k = \lfloor K/L \rfloor$ blocks, $X_1, \ldots, X_k$. Next, for each $i$, we multiply $X_i$ by the inverse of the $i$-th $L \times L$ submatrix of $G$. Each of these produces a candidate $s_i$ for the seed. If $X$ is within a distance of $\lfloor K/2L \rfloor$ from any valid codeword $Z$, at least $\lfloor K/2L \rfloor$ of the $k$ $s_i$'s must agree and must equal the seed $s$ that produced $Z$. It is possible that $X$ is not within a small distance from any valid codeword, yet more than half of the $s_i$'s agree. To check this, the popular $s$ can be multiplied by $G$ to see if it produces a codeword that is close to $X$. Clearly, this can be done in time polynomial in $N$. We summarize the properties of this enhanced generator in the lemma below:

**Lemma 4.4** *The generator of Theorem 4.1 can be designed to satisfy the following additional properties:*

(1) *For every odd $k$, $1 \leq k \leq \lfloor N/L \rfloor$, the range of the generator, when restricted to the first $kL$ bits, is a linear code of minimum distance at least $k$; thus it can correct up to $\lfloor k/2 \rfloor$ errors.*

(2) *Given any string $X$ of length $N$ and any odd $k \leq \lfloor N/L \rfloor$, we can determine, in time polynomial in $N$, if there is some seed $s \in \{0,1\}^L$ such that the Hamming distance, within the first $kL$ positions, between $X$ and $sG$ is at most $\lfloor k/2 \rfloor$. Moreover, if such an $s$ exists, it can also be identified in the same time.*

(3) *The construction of the generator matrix $G$ can be done in time polynomial in $N$.*

## 4.4 Non-$\Gamma_\kappa(\mathrm{P})$-measurability of $\mathrm{AC}^0_4[\oplus]$

In this section, we build on techniques from the previous chapter, particularly the insight behind the proof of Theorem 3.3, and prove an analogous result at the polynomial-time setting, using Nisan's pseudorandom generator in lieu of the hypothetical generator in Theorem 3.3.

**Theorem 4.5** $\mu^*_{\Gamma_\kappa(\mathrm{P})}(\mathrm{AC}^0_4[\oplus]) \neq 0$. *Specifically, for every $\Gamma_\kappa(\mathrm{P})$-martingale $\kappa$, there is a language $L \in \mathrm{AC}^0_4[\oplus]$ (nonuniform) on which $\kappa$ does not become unbounded.*

We begin with a slightly improved version of Lemma 3.4. The difference here is that the capital is shown to be bounded at all points from $u$ to $uv$, not merely at the endpoints.

**Lemma 4.6** *Let $\kappa$ be a martingale. For any string $u$ and any $\ell \in \mathbb{N}$, $r \in \mathbb{R}$,*

$$\left\| \{v \in \{0,1\}^\ell : (\forall w \sqsubseteq v)\kappa(uw) \leq \left(1 + \frac{1}{r}\right)\kappa(u)\} \right\| \geq 2^\ell \left(\frac{1}{r+1}\right).$$

**Proof.** It follows from the definition of a martingale that $\sum_{v\in\{0,1\}^\ell} \kappa(uv) = 2^\ell \cdot \kappa(u)$. Suppose by way of contradiction that for some $u, \ell$, and $r$, the inequality in the statement of the lemma does not hold. This implies

$$\left\|\{v \in \{0,1\}^\ell : (\exists w \sqsubseteq v)\kappa(uw) > \left(1+\frac{1}{r}\right)\kappa(u)\}\right\| \geq 2^\ell\left(\frac{r}{r+1}\right). \tag{4.1}$$

Consider the complete binary tree $T$ of depth $\ell$, and imagine that the root of the tree is endowed with a capital sum of $\kappa(u)$. The interior nodes of $T$ can be associated in one-to-one correspondence with $\{0,1\}^{<\ell}$, and the leaves of $T$ with $\{0,1\}^\ell$ in the obvious way; we will, therefore, refer to the nodes of $T$ directly as strings in $\{0,1\}^{\leq\ell}$. Each node $v$ in the tree (leaves as well as interior nodes) will be annotated by the value $\kappa(uv)$. The annotations describe the *strategy* of martingale $\kappa$ in the obvious way. Call an interior node $w$ of $T$ *rich* if $\kappa(uw) > (1+1/r)\kappa(u)$, and call a leaf $v$ of $T$ *prodigal* if $v$ is the descendant of some rich interior node $w$. Then Equation 4.1 is the same as saying that the number of prodigal leaves $v$ is greater than $2^\ell \cdot (r/r+1)$.

For each prodigal leaf $v$, mark the rich ancestor of $v$ that is closest to the root. It is clear that along any path from the root of $T$ to any leaf, there can be at most one rich node. Now for each rich node $w$, annotate the entire subtree of $w$ by the value $\kappa(uw)$. Let $T'$ denote the resulting tree. It is easy to see that $T'$ represents a valid martingale strategy, since annotating the subtree of node $w$ by $\kappa(uw)$ corresponds to playing a safe strategy (without making any wager) on these strings. More importantly, we claim that every leaf $v$ that was labeled *prodigal* in $T$ is *rich* in $T'$. To see this, note that if $v$ was prodigal in $T$, some rich ancestor $w$ of $v$ was marked, and later $v$ inherits all the wealth of $w$, so $v$ is rich.

Let $\kappa'$ denote the martingale that behaves exactly like $\kappa$ on all strings of length at most $|u|$, and then adopts the strategy given by $T'$ on extensions of $u$. Under this strategy, the number of rich leaves in $T$ is at least $2^\ell \cdot (r/r+1)$, and each rich leaf has an annotated value of strictly greater than $(1+1/r)\kappa(u)$. Therefore, the total money that $\kappa'$ has at the bottom of $T'$, namely, $\sum_{v\in\{0,1\}^\ell} \kappa(uv)$, exceeds $2^\ell$, a contradiction. □

To put the idea of Lemma 4.6 to concrete use, we will use the statistical test of Figure 4.2, which we denote by $T_{\kappa,u}$ (to denote that this test is derived from the martingale $\kappa$, and that it has the string $u$ hardwired into it).

```
Algorithm $T_{\kappa,u}$, where $|u| = 2^n - 1$:
Input $v \in \{0,1\}^{2^n}$
Accept if and only if $(\forall w \sqsubseteq v)\kappa(uw) \leq (1+1/n^2)\kappa(u)$.
```

Figure 4.2: Statistical Test $T_{\kappa,u}$

We begin by analyzing the complexity of implementing the statistical test $T_{\kappa,u}$. Let $f(m) = (\log m)^c$ bound the running time and dependency set size of a Turing Machine that computes the martingale $\kappa$. On inputs of length $2^n$, $f(2^n) = n^c$. For each $v \in \{0,1\}^{2^n}$ and each $w \sqsubseteq v$, we can pre-compute the predicate "$\kappa(uw) \leq (1+1/n^2)\kappa(u)$." Since the computation of $\kappa(uw)$ depends on only $n^c$ bits of $w$, this predicate can be described by a truth table of size $2^{n^c}$. By hardwiring this truth table as a sum-of-products, we get a circuit of size $2^{n^c}$ and depth 2 using only AND, OR, and NOT gates. Finally, the statistical test $T_{\kappa,u}$ may be implemented by computing the predicate "$(\forall w \sqsubseteq v)[\kappa(uw) \leq (1+1/n^2)\kappa(u)]$," which can be computed by taking the AND of each of the above $2^n$ circuits. Thus, $T_{\kappa,u}$ can be computed by a depth-3 circuit on $2^n$ inputs that has only AND, OR, and NOT gates, and that has size at most $2^{O(n^c)}$.

How likely is $T_{\kappa,u}$ to accept a random string $v$ of length $2^n$? By Lemma 4.6, it is clear that for any $u$ of length $2^n - 1$, the probability, when a random $v$ is chosen from all strings of length $2^n$, that $T_{\kappa,u}(v)$ accepts is at least $1/n^2$. Here is where a pseudorandom generator enters the picture: what if we randomly choose a string from the range of a pseudorandom generator that produces $2^n$ bits, and that is provably secure against all circuits of size $2^{n^c}$ and depth 3 that have $2^n$ inputs? If the generator is good, then the probability that $T_{\kappa,u}$ accepts when the input $v$ is chosen from the range of the generator must be very close to $1/n^2$. In particular, if the generator is strong enough, we can guarantee that there is *at least one string* $v$ in the range of the generator that $T_{\kappa,u}$ must accept. It is now amply clear that the algorithm of Figure 4.3 produces a language $A$ on which $\kappa$ stays bounded.

```
Let $n_0$ be the first integer such that $N_0 = 2^{n_0}$ is large enough
    to apply Nisan's construction.
Arbitrarily fix the memberships in $A$ of all strings of length less than $n_0$.
Set $u \leftarrow A^{<n_0}$
Set $n \leftarrow n_0$
while TRUE do:
    Let $G_n$ denote the instance of Nisan's generator from Theorem 4.1
        that is secure against circuits with $2^n$ inputs of size $2^{O(n^c)}$ and depth 3.
    Let $v \in \{0,1\}^{2^n}$ be the first string in the range of $G_n$ that $T_{\kappa,u}$ accepts.
    Set $A^{=n} \leftarrow v$.
    Set $u \leftarrow u \circ v$ (here $\circ$ denotes concatenation).
    Set $n \leftarrow n+1$.
```

Figure 4.3: Algorithm to produce a language that defeats a given martingale $\kappa$

What is the complexity of the language $A$ defined by the algorithm of Figure 4.3? The parameters needed in the application of Theorem 4.1 in the definition of $A$ are as follows: $N = 2^n$, $D = 3$, and $2^{\frac{1}{10}M^{1/(4+\varepsilon)}} = 2^{O(n^c)}$. It suffices, therefore, to take $M = O(n^{(4+\varepsilon)c})$ for some $\varepsilon > 0$. From the construction of Nisan's generator, $L = M^2 = O(n^{9c})$. From Fact 4.3, we know that $A$ is accepted by a family of circuits of size $(n+L)^{O(1)} = n^{O(1)}$ and depth 4 that uses only AND and PARITY gates, whence it follows that $A \in \mathrm{AC}_4^0[\oplus]$. Theorem 4.4 is proven. □

## 4.5 AC$^0$ has $\Gamma_\beta(\mathrm{P})$-Measure Zero

We earlier alluded to a result of Allender and Strauss [AS94a] that places a limit on the power of a $\Gamma_\kappa(\mathrm{P})$-martingale that succeeds in the sense of a limit: for any such martingale, there is a sparse language (which, by virtue of being sparse, is also in depth-2 AC$^0$) on which the martingale does not succeed. We remarked that there is a $\Gamma_\kappa(\mathrm{P})$-martingale that nevertheless succeeds on SPARSE in the sense of lim sup. The main result of the foregoing section places a limit on the power of $\Gamma_\kappa(\mathrm{P})$-martingales, even with success defined as becoming unbounded. We showed that for every such martingale, there is a language $L \in \mathrm{AC}^0_4[\oplus]$ on which the martingale stays bounded. It didn't seem possible to place the language $L$ into AC$^0$ itself. It is not clear also if such a martingale can be built to succeed on all of AC$^0$, or even on depth-2 AC$^0$.

In this section, we we examine the power of $\Gamma_\beta(\mathrm{P})$-martingales, which are seemingly more powerful than their $\Gamma_\kappa$ brethren. The difference in the power of the two types of martingales is not very clear, however; after all, even the $\Gamma_\beta(\mathrm{P})$-martingales must be computable in time $\log^{O(1)}|w|$ on an input $w$, and must satisfy the stringent conditions on the size of the dependency set. How much more power do we obtain by requiring the martingale to only output the bet amount $\beta(w)$ rather than the capital amounts $\kappa(w0)$ and $\kappa(w1)$? Surely if the martingale bets \$500 on the membership of the next string $x$, it knows that at least \$500 is available at the node $w$, i.e., that $\kappa(w) \geq \$500$. How much more can it possibly achieve than a $\Gamma_\kappa(\mathrm{P})$-martingale that attempts to succeed in the sense of a lim sup? We prove the following result that is somewhat surprising in the light of these remarks:

**Theorem 4.7** *For an integer $d$, let $\widehat{\mathrm{AC}^0_d}$ denote the collection of all languages that are recognized by nonuniform families of circuits with AND, OR, and NOT gates of depth $d$ and size $2^{n^{\frac{1}{2d+6+\delta}}}$ (for all sufficiently large $n$ and for any fixed $\delta > 0$). For any $d = o(\log n/\log\log n)$, there is a $\Gamma_\beta(\mathrm{P})$-martingale that succeeds on all of $\widehat{\mathrm{AC}^0_d}$ (in the sense of a limit).*

Since $\widehat{\mathrm{AC}^0_d}$ includes nondeterministic time $O(n^{1/10+o(1)})$ (with $d = 2$) and alternating time $\log^{O(1)} n$ (with $d = \omega(1)$), we obtain the following corollaries:

**Corollary 4.8** $\mu_{\Gamma_\beta(\mathrm{P})}(\mathrm{NTIME}[n^{1/10+o(1)}]) = 0$.

**Corollary 4.9** $\mu_{\Gamma_\beta(\mathrm{P})}(\mathrm{ATIME}[\log^{O(1)} n]) = 0$.

Although Theorem 4.7 is the strongest statement, Corollaries 4.8 and 4.9 have their own independent interest in resource-bounded measure theory. Recall that one of the central hypotheses in resource-bounded measure theory is the hypothesis that NP does not have measure zero in EXP. The main reason for disbelieving the hypothesis $\mu(\mathrm{NP}|\mathrm{EXP}) = 0$ is that a martingale computed by a fixed exponential-time, say $2^{n^c}$, deterministic Turing machine cannot search through search spaces of size $2^{n^d}$, $d > c$, that arise from nondeterministic time $n^d$ machines. This hypothesis, when scaled down, translates to the hypothesis that NTIME$[(\log n)^{O(1)}]$ does not have measure zero in quasipolynomial time QP. Of course, the machine model for defining the class NTIME$[(\log n)^{O(1)}]$ is different from the one in which NP is defined. Nevertheless, the reasoning outlined above holds just as well here:

how can a single quasipolynomial-time Turing machine running in time $2^{\log^c n}$ search through spaces of size $2^{\log^d n}$ for arbitrary $d$ and diagonalize against all such languages? Corollary 4.8 says that such algorithms exist; in fact, ones that are nearly exponentially more powerful exist.

Our proof techniques in this Section are based primarily on circuit complexity theory and pseudorandom generators. The main ingredients consist of Håstad's Switching Lemma [Hås89], Nisan's pseudorandom generator [Nis91, NW88], and error-correcting codes. We recall from Section 4.3 that Nisan's pseudorandom generator is secure against constant depth circuits, and the proof of *that* depends on the lower bound for constant depth circuits, particularly the decision tree version of Håstad's Switching Lemma.

OVERVIEW OF THE PROOF. For the class of constant depth circuits of the appropriate size bound, we first apply Håstad's Switching Lemma to conclude that circuits would have been significantly "demolished" by a random partial assignment to the inputs, i.e. a *random restriction*. However these truly random restrictions will be of little use to our martingale, since they don't possess enough structure to meet the dependency set requirements, etc. We then apply Nisan's pseudorandom generator to substitute the truly random restrictions by pseudorandom restrictions, and argue that the pseudorandom restrictions from Nisan's pseudorandom generator work almost as well as truly random restrictions. The basic argument is that, if there were a circuit $C$ whose behavior under pseudorandom restrictions is significantly different from that under true random restrictions, then this $C$ would have been a statistical test that Nisan's generator would not pass; but the fact is that Nisan's generator *is* secure, and no such statistical test exists. We note that the very reason for the security of Nisan's generator is the strong decision-tree lower bound againt constant depth circuits that follow from (a modification of) Håstad's Switching Lemma. Thus, curiously, the Switching Lemma enters our proof in two successive steps: First, to argue that under a random restriction the circuit is likely to be killed; second, if we replaced the true random restriction by a pseudorandom restriction, the circuit is also likely to be killed, because it couldn't tell the difference—and the reason why it could not tell the difference is precisely, again, (the decision tree version of) the Switching Lemma.

The pseudorandom restrictions obtained directly from Nisan's generator turn out to be not quite good enough for the purpose of constructing our martingales. For that purpose, we need some additional structures, especially error-correcting codes. To this end, we use the enhancement of Nisan's generator that we sketched in Lemma 4.4. Using this construction, the pseudorandom restrictions coming out of the modified generator not only are pseudorandom, but they also form a certain error-correcting code, which makes the construction of our martingales possible.

The martingale will "defeat" all constant depth circuits of an appropriate size, not by seeking out individually each circuit, of which there would have been too many—precisely the argument *in favor of* the Lutz hypothesis—but by killing such circuits *collectively* and *obliviously*.

In Section 4.5.1 we summarize background material on the Switching Lemma for boolean circuits and prove a technical lemma based on the Switching Lemma. In Section 4.5.2 we show the construction of the $\Gamma_\beta(\mathrm{P})$-martingale that defeats all of $\widehat{\mathrm{AC}^0_d}$.

### 4.5.1 Collapsing Circuits by Random Restrictions

Let us denote the inputs of a Boolean circuit by the variables $x_i$ and $\overline{x_i}$ (the latter represents the negation of $x_i$). We may assume that the circuit is *leveled*, i.e., whether a gate is an AND or an OR is determined by the parity of its distance from the input.

Let $C_n$ be a leveled circuit with $n$ inputs $x_1, x_2, \ldots, x_n$. A *restriction* (of the inputs of $C_n$) is a partial assignment $\rho_n : \{x_1, \ldots, x_n\} \to \{0, 1\}$. If $\rho_n(x_i)$ is undefined we write $\rho_n(x_i) = \star$. A *random restriction of strength p* is a restriction $\rho_n$ obtained by independently setting each $x_i$ to $\star$ with probability $p$, and to 0 or 1 with probability $(1-p)/2$. We denote by $C_n|\rho_n$ the circuit resulting from instantiating the inputs of $C_n$ according to $\rho_n$, when the latter is defined.

**Lemma 4.10 (Håstad[Hås89])** *Let $C_n$ be an AND of OR's with n inputs and bottom fan-in at most $t$, and let $\rho$ be a random restriction of $C_n$ of strength p. Then, for all integers $s \geq 0$,*

$$\Pr[C_n|\rho \textit{ has a minterm of size } \geq s] \leq (\alpha p t)^s,$$

*where* $\alpha \approx 4.9 < 5$.

That is, if an AND of OR's of small bottom fan-in is hit with a random restriction, then with high probability, the result is equivalent to an OR of AND's of small bottom fan-in. A similar statement applies to switching an OR of AND's to an AND of OR's.

The following technical lemma will play a crucial role in the construction of our martingale. The twist here is that the restrictions are not chosen uniformly at random of some strength, but are chosen from the special family $\mathcal{F}_{2\log n}$ of all restrictions that have exactly $2\log n$ stars. We prove that in this case, with reasonable probability, the circuit collapses to a constant.

**Lemma 4.11** *Let $d \geq 2$ be an integer, let $\delta > 0$, and let $\varepsilon = 1/(d+\delta)$. Let C be a circuit of depth d and size at most $2^{\frac{1}{10}n^{\varepsilon}}$ with AND, OR, and NOT gates, that has n inputs. Let $\rho$ be a restriction chosen uniformly from $\mathcal{F}_{2\log n}$, i.e., from among restrictions that have exactly $2\log n$ stars. Then*

$$\Pr[C|_\rho \textit{ is not a constant}] = o(1).$$

**Proof.** Wlog. we may assume that the circuit $C$ consists of alternating levels of AND and OR gates. By adding an extra dummy row of gates of fan-in 1 at the bottom of the circuit and increasing the depth to $d+1$, we may assume that $C$ has bottom fan-in 1. These dummy gates will be chosen to be AND gates if $C$ consists of only OR gates at the first level, and vice-versa. This ensures that after applying a random restriction, if we succeed in switching the lowest two levels of the circuit, then we can collapse the second and third layers (which, after the switching, must be of the same type).

We will prove the lemma in two stages. First we will describe a probabilistic process where the circuit successively undergoes $d$ random restrictions of a certain strength. We will argue that whp. we have at least $2\log n$ input variables that remain unassigned, and moreover, the circuit will collapse to a constant. Then we will argue that subjecting the circuit to a restriction uniformly chosen among all restrictions with exactly $2\log n$ stars produces essentially the same effect as the probabilistic process outlined in the first stage.

Let $t = \frac{1}{10}n^{\varepsilon}$. We further sub-divide the first stage into two steps. First we apply $d-1$ random restrictions of strength $p = \frac{1}{n^{\varepsilon}}$. During the application of any of these $d-1$ random restrictions, with exceptional probability $<$ size of the circuit $\times$ $(\alpha pt)^s$, we succeed in switching every depth-2 subcircuit and leaving the bottom fan-in $\leq s$. This exceptional probability equals $2^{\frac{1}{10}n^{\varepsilon}}(\alpha(1/n^{\varepsilon})\frac{1}{10}n^{\varepsilon})^s$. Taking $s = \frac{1}{10}n^{\varepsilon}$, this exceptional probability is at most $\beta^s$, where $\beta = 2\alpha/10 \approx 0.98$, and the probability that we fail to perform switching in any one of the $d-1$ random restrictions is at most $(d-1)\beta^{\frac{1}{10}n^{\varepsilon}}$. Therefore with this exceptional probability, we will be left with a depth-2 circuit of bottom fan-in at most $\frac{1}{10}n^{\varepsilon}$.

We continue the probabilistic process by subjecting the remaining circuit to a random restriction of strength $q = \frac{1}{n^{(1+\delta/2)\varepsilon}}$, and invoke the switching lemma with $s = 1$. This implies that with exceptional probability at most $\alpha qt < \frac{1}{2n^{\frac{\delta}{2(d+\delta)}}}$, the circuit collapses to a constant. Notice, moreover, that the probability that any particular input variable remains unassigned (i.e. remains a star) is precisely $\gamma = p^{d-1}q = 1/n^{(\frac{d+\delta/2}{d+\delta})}$, independent of all the other variables. Hence the expected number of variables that remain unassigned is $np^{d-1}q = n^{\delta/(2(d+\delta))}$, which is much more than $2\log n$, so long as $d$ is much less than $\delta \log n/2\log\log n$. For large enough $n$, the probability that strictly fewer than $2\log n$ input variables remain is

$$\begin{aligned} &= \sum_{i=0}^{2\log n-1}\binom{n}{i}\gamma^i(1-\gamma)^{n-i} \\ &\leq 2\log n\binom{n}{2\log n}(1-\gamma)^{n-2\log n} \\ &\leq 2^{O(\log n)^2}e^{-n^{\frac{\delta/2}{d+\delta}}}. \end{aligned}$$

We will finish the probabilistic process by randomly selecting $2\log n$ of the remaining stars, and assigning the other stars to 0 or 1 with equal probability. Thus with high probability, our probabilistic process collapses the circuit to a constant, and retains exactly $2\log n$ stars.

To complete the lemma, we must establish that $C$ collapses to a constant with essentially the same probability if we uniformly pick a restriction from the family $\mathcal{F}_{2\log n}$ that consists of all restrictions on $n$ variables with exactly $2\log n$ stars. The process of uniformly choosing a restriction from $\mathcal{F}_{2\log n}$ can be *realized* by the following two-step process: First run our probabilistic process of successive restrictions described above. If it produces fewer than $2\log n$ stars, report failure. Otherwise, use the resulting restriction from our probabilistic process. Notice that when this two-step process does succeed in producing $2\log n$ stars, it produces every restriction in $\mathcal{F}_{2\log n}$ with equal probability. Therefore,

$$\begin{aligned} &\Pr_{\rho\in\mathcal{F}_{2\log n}}[C|_{\rho} \text{ does not collapse to a constant}] \\ \leq\ &\Pr[\text{our probabilistic process produces fewer than } 2\log n \text{ stars}] \\ +\ &\Pr[C \text{ does not collapse to a constant} \mid \text{our probabilistic process produces} \geq 2\log n \text{ stars}] \end{aligned}$$

$$
\begin{aligned}
&\le \quad \frac{2^{O(\log n)^2}}{e^{n^{\frac{\delta/2}{d+\delta}}}} + (d-1)\beta^{\frac{1}{10}n^{\frac{1}{(d+\delta)}}} + \frac{1}{2n^{\frac{\delta}{2(d+\delta)}}} \\
&= \quad o(1).
\end{aligned}
$$

□

We call a circuit *collapsible* if, with probability $1 - o(1)$, a random restriction having exactly $2\log n$ stars collapses the circuit to a constant.

## 4.5.2 Construction of the Martingale

OVERVIEW OF THE CONSTRUCTION. We seek to construct a $\Gamma_\beta(\mathrm{P})$-martingale that covers the class of languages recognized by collapsible circuits. For definiteness, let us suppose that the martingale wishes to defeat all depth $d$, size $2^{\frac{1}{10}n^{\frac{1}{2d+6+\delta}}}$ circuits with AND, OR, and NOT gates, for a small $\delta > 0$.

The martingale we construct will reserve $\Theta(1/n^2)$ capital to bet on words of length $n$, so that we can simply focus on an arbitrary stage $n$, and demonstrate that the martingale makes huge profits at all stages $n$, for sufficiently large $n$, along any such language.

The first attempt that illustrates our basic strategy is as follows: the martingale will be designed to make huge profits on all boolean functions on $n$ variables that are collapsed to a constant by a random restriction with exactly $2\log n$ stars. Since the latter class contains all boolean functions recognized by constant-depth nearly exponential-size circuits, the martingale will make a huge profit on every such function. To this end, the martingale $\beta$ will consist of $2^{\binom{n}{2\log n}+1}$ "shares" $\beta_{\rho,b}$, where $\rho$ is a restriction with exactly $2\log n$ stars, and $b \in \{0,1\}$, and $\beta_{\rho,b}$ covers all circuits that are collapsed to the constant $b$ under the restriction $\rho$. The main point, of course, is that every function recognized by a constant-depth nearly exponential-size circuit *must* then be covered by at least one share $\beta_{\rho,b}$.

There are two main problems with this strategy. First, the number of $(\rho,b)$ pairs is far too large for a polynomial time martingale to evaluate them all and compute their sum. This problem can be surmounted by noting that for any word $x$, there are exactly $\binom{n}{2\log n}$ restrictions $\rho$ that are compatible with $x$, hence only $2\binom{n}{2\log n} = O(n^{\log n})$ shares are *relevant*, and all of these can be precisely identified from $x$. This would give us a quasipolynomial time strategy, but it still suffers from a second and the more serious problem: for any $x$, while the dependency set $G_x$ of $x$ is small (quasipolynomial), the property "$(\forall y \in G_x) G_y \subset G_x$" (see Section 4.1) is violated.

We solve both problems by designing a suitable family $\mathcal{R}$ of *pseudorandom restrictions* that satisfies the following properties: (1) there will be at most $2^n$ restrictions in $\mathcal{R}$ so that each share can start with a capital of $\Omega(2^{-n}/n^2)$, (2) most of the restrictions will have exactly $2\log n$ stars, so that the corresponding shares will have $n^2$ distinct occasions to double their capital so as to multiply their puny $O(2^{-n}/n^2)$ capital by $2^{n^2}$ for a huge return, (3) given any word $x$, there will be at most one restriction $\rho \in \mathcal{R}$ that is compatible with $x$, and finally, (4) every function recognized by a depth-$d$ size-$2^{\frac{1}{10}n^{1/(2d+6+\delta)}}$ circuit will be collapsed to a constant by at least one restriction in $\mathcal{R}$.

THE FAMILY $\mathcal{R}$ OF PSEUDORANDOM RESTRICTIONS. To construct the family $\mathcal{R}$, we will use our enhancement of Nisan's generator [Nis91] that we presented in Lemma 4.4. With respect to Theorem 4.1, we will set $N = n + 2\log^2 n$, $D = d + 2$, and $L = n/(4\log n + 1)$, and apply Lemma 4.4 with $k = 4\log n + 1$ (so that $kL = n$). The collection of strings generated by this enhanced generator not only looks pseudorandom to shallow circuits, but also has a minimum distance $\Delta = 4\log n + 1$ in the first $n$ positions between every pair of codewords.

The output of the generator will be a string of length $n + 2\log^2 n$, which is interpreted as a restriction by assigning the first $n$ bits to the $n$ variables, and then interpreting the $2\log^2 n$ remaining bits as the positions of $2\log n$ stars (the stars replace some bits at first assigned). We endow the first $n$ bits of the codewords with minimum distance $\Delta$ so that for each $x$ there is at most one codeword $z$ whose distance from $x$ is within $2\log n$, and we can find the codeword quickly (by decoding). It is also easy to check if this codeword $z$, when interpreted as a restriction $\rho_z$, is compatible with $x$ with respect to the non-star positions. Finally, from the details in Sections 4.3 and Section 4.4, it is clear that the range of the generator has at most $2^L \leq 2^n$ codewords, and that this entire construction can be carried out in time polynomial in $n$.

Next, we claim that given any circuit $C$ of size $2^{\frac{1}{10}m^{\frac{1}{(d+3+2\varepsilon)}}}$ and depth $d$, there is at least one restriction $\rho_z$ produced from some codeword $z$ such that (1) $\rho_z$ has exactly $2\log n$ stars, and (2) $\rho_z$ collapses $C$ to a constant. Suppose this is not the case for some circuit $\widehat{C}$. Then we claim that the statistical test $T$ described in Figure 4.4 distinguishes the output of the modified Nisan's generator from a truly random string of length $N = n + 2\log^2 n$.

```
input $z \in \{0,1\}^{n+2\log^2 n}$
output 1 iff
    The last $2\log^2 n$ bits of $z$,
        interpreted as $2\log n$ numbers in the range $[1,n]$, are all distinct
    and
    when $z$ is interpreted as a restriction $\rho_z$,
        $(\exists b \in \{0,1\})$ $(\forall x$ compatible with $\rho_z)$ $[\widehat{C}(x) = b]$.
```

Figure 4.4: Statistical Test $T$ for the modified Nisan generator

We proceed to analyze the statistical test $T$. The probability that a truly random input of $n + 2\log^2 n$ bits will be accepted is quite high: By Lemma 4.11, a random input will fail the second part of the test with probability at most $o(1)$. The probability of producing a collision in $2\log n$ choices of numbers in the range $[1,n]$ is at most $\binom{2\log n}{2}(1/n) \leq 2\log^2 n/n$, so the first test also fails with probability at most $2\log^2 n/n$. Thus a random input passes the test $T$ with probability at least $1 - o(1)$.

On the other hand, suppose that all the $\rho_z$'s produced from the modified Nisan generator fail the test $T$ (recall that our goal is just to show that at least one of the $\rho_z$'s will pass the test $T$, rather than prove stronger bounds on the probability of a $\rho_z$ passing the test). Then clearly $T$ achieves a bias of $1 - o(1)$ in distinguishing the output of the generator from a truly random string.

Finally, we note that the statistical test $T$ can be implemented by a circuit of depth $d + 2$ and size $n^2\binom{n}{2\log n}S$, where $d$ is the depth of $\widehat{C}$ and $S$ is the size of $\widehat{C}$. Performing the first part of the test can be done by an AND of OR's of size $2^{2\log^2 n}$. Multiple copies of this test will be AND'ed, in parallel, into the second part at an appropriate level. For the second part, we create $n^2\binom{n}{2\log n}$ copies of $\widehat{C}$, each corresponding to a pair $(U,v)$, where $U \subseteq [1,n], |U| = 2\log n$, specifies $2\log n$ star positions and $v \in \{0,1\}^{2\log n}$ specifies an instantiation of the $2\log n$ stars. Precisely one of the $U$'s will be "enabled" by the last $2\log^2 n$ bits of the input $z$ in the obvious way; this can be done by a single AND gate that takes $2\log^2 n$ inputs (positive and negated input bits), and in parallel with the rest of the circuit, so it doesn't count toward depth. We will take an OR over all the different pairs of $U$ with a bit. For each $U$ and each bit $b$, each of the $n^2$ circuits labeled by $v \in \{0,1\}^{2\log n}$ fills in the star positions in one of the $n^2$ possible ways, retains the other bits of $z$, and computes the output of $\widehat{C}$ on this. Checking if all the $n^2$ circuits produce $b$ (i.e. if $\widehat{C}$ collapsed to $b$) can be done by a single AND (if $b = 0$ first negate $\widehat{C}$ and apply De Morgan). By collapsing adjacent levels of the same type, the statistical test can be implemented by a circuit that has depth at most $d+2$, and size at most $n^2\binom{n}{2\log n}2^{\frac{1}{10}m^{\frac{1}{(d+3+2\varepsilon)}}} \le 2^{\frac{1}{10}m^{\frac{1}{(d+3+\varepsilon)}}}$, contradicting the security of the modified generator against such circuits. Taking $\delta = 2\varepsilon$, we have:

**Theorem 4.12** *For every $\delta > 0$, every integer $d \geq 2$ and every sufficiently large integer $n$, there is a family $\mathcal{R}$ of pseudorandom restrictions such that:*

*(1) $|\mathcal{R}| \le 2^{\frac{n}{4\log n+1}} \le 2^n$.*

*(2) For any word $x \in \{0,1\}^n$, there is at most one restriction $\rho \in \mathcal{R}$ that is compatible with $x$, and there is a uniform way of identifying $\rho$ from $x$ in time polynomial in $n$.*

*(3) For any $n$-input circuit $C$ of depth $d$, size $2^{\frac{1}{10}n^{\frac{1}{2d+6+\delta}}}$ with AND, OR, and NOT gates, there is at least one $\rho \in \mathcal{R}$ that has exactly $2\log n$ stars and that collapses $C$ to a constant.*

THE MARTINGALE. We are now ready to describe our martingale that defeats circuits of size $2^{\frac{1}{10}n^{1/(2d+6+\delta)}}$ of depth $d$. The martingale allocates roughly $1/2n^2$ as the "capital" for each $n$ ($\sum_n 1/2n^2 < 1$), and starts making its bets from a sufficiently large stage $n$ (so that Theorem 4.12 applies).

For each $n$, the martingale is composed of shares $(\rho, b)$ for each pseudorandom restriction $\rho \in \mathcal{R} = \mathcal{R}(n,d,\delta)$ (from Theorem 4.12) and each $b \in \{0,1\}$. Each share $(\rho,b)$ starts with a capital of $2^{-n}/2n^2$, and for every word $x$ compatible with $\rho$, bets all of *its* available capital that $x$ has an "answer" $b$. Whether or not any particular bet succeeds has no meaning in itself; the point is, given any boolean function $f$ (equivalently, a segment of a language) that is recognized by a depth $d$ nearly exponential-size circuit, there is at least one restriction $\rho \in \mathcal{R}$ that collapses $f$ to either 0 or 1, and the corresponding share doubles its capital as many times as there are words consistent with it.

Formally, let the input to the martingale be $w$, $2^n \le |w| < 2^{n+1}$. Recall that such an input $w$ is interpreted as specifying the memberships of all words that precede some $x$ of length $n$. The output of the martingale should be its bet on the membership of $x$. The martingale first performs the decoding of the modified Nisan generator and identifies the unique restriction $\rho$, if one exists, that is compatible with $x$. If none exists, then the martingale outputs a bet of 0 on $x$. If $\rho$ is compatible with $x$, then the martingale "activates" the two shares $(\rho,0)$ and $(\rho,1)$. The share $(\rho,b)$ first produces the list of all words $x_1, x_2, \ldots, x_k$ preceding $x$ that are compatible with $\rho$. Notice that this set of words is precisely the dependency set $G_x$ of $x$, and this share would have been activated on precisely those words in $G_x$.

It is clear that $|G_x| \leq n^2$. Furthermore, the share $(\rho, b)$ probes precisely those bits of $w$ that give the memberships of $x_1, \ldots, x_k$, and checks if all those bits of $w$ are precisely $b$. Figuratively, this means that this share $(\rho, b)$ would not only have been activated for those words, but along the path $w$, this share would have won all its bets, thus taking its capital from $1/(2n^2 2^n)$ to $2^k/(2n^2 2^n)$. If these tests are satisfied, the share $(\rho, b)$ bets all its earnings, namely $2^k/(2n^2 2^n)$ that the membership of $x$ is $b$.

A minor point worth noting: one might wonder that the two shares $(\rho, 0)$ and $(\rho, 1)$ always make opposite bets and cancel each other out. If $x$ is the first word of length $n$ that is compatible with $\rho$, this is indeed the case, and the bet on $x$ would just be 0. However, if $x$ is not the first word of length $n$ that is compatible with $\rho$, then one of the two shares $(\rho, 0)$ and $(\rho, 1)$ must fail the "history test" with respect to the given input $w$, so the martingale's bet on $x$ is simply the bet of the winner $(\rho, b)$ (assuming the winner passes the tests).

More important are the following facts: the running time of the martingale is polynomial in $n$, and the dependency sets are of size at most $n^2$ and satisfy the condition $(\forall y \in G_x) G_y \subset G_x$. Furthermore, by Theorem 4.12, it follows that for inputs $w$, $|w| = 2^{n+1}$, such that the last $2^n$ bits of $w$ represent a boolean function $f$ that can be recognized by a depth $d$ size $2^{\frac{1}{10} n^{\frac{1}{2d+6+\delta}}}$ circuit, at least one of the shares would take its capital to $2^{n^2}/(2n^2 2^n)$, which is unbounded. The proof of Theorem 4.7 is complete.

□

## 4.6 Non-$\Gamma_\beta(\mathrm{P})$-measurability of $\mathrm{AC}^0_4[\oplus]$

The result of the previous section established that $\Gamma_\beta(\mathrm{P})$-martingales are quite powerful, presumably more powerful than a $\Gamma_\kappa(\mathrm{P})$-martingale that succeeds by lim sup, since the latter class of martingales seem to be inadequate to cover $\mathrm{AC}^0$. It might be tempting to believe that $\Gamma_\beta(\mathrm{P})$-martingales can also cover $\mathrm{AC}^0[\oplus]$, a class that $\Gamma_\kappa(\mathrm{P})$-martingales couldn't cover even in the lim sup sense. However, in this section, we prove that this belief is false.

In fact, we show that the same pseudorandom generator that we used in Section 4.4 (with essentially the same settings of parameters) will also give us a language on which $\Gamma_\beta(\mathrm{P})$-martingale does not succeed in the limit. Very curiously, however, the proof technique that worked in Section 4.4 does not work anymore! The structure of the proof is the same, but the fact that Nisan's generator is secure against constant-depth circuits seems to be of little use here; instead, we need to exploit the additional properties of Nisan's generator that we established in Section 4.3.

Let $\kappa$ denote the$\Gamma_\beta(\mathrm{P})$-martingale that we are attempting to defeat; let $\beta$ denote the betting function derived from $\kappa$. Let $c$ be a constant such that, for any $w$, $\beta(w)$ is computable in time $\log^c |w|$. As in Section 4.4, we begin by describing a statistical test $T_{\beta,u}$, which we present in Figure 4.5.

We begin with the question: Is the analogue of Lemma 4.6 true for the test $T_{\beta,u}$? i.e., is it true that $T_{\beta,u}$ accepts a randomly chosen $v$ with about $O(1/n^2)$ probability? Indeed it is, since $\kappa$ is a martingale induced by the betting function $\beta$, and Lemma 3.4 guarantees (by Markov inequality) that $T_{\beta,u}$ accepts a random $v$ with probability at least $1/n^2$. Thus, it would suffice to find a smaller sample space from which we can sample the $v$'s, and ensure that $T_{\beta,u}$ would accept with essentially the same probability.

```
Algorithm $T_{\beta,u}$, where $|u| = 2^n - 1$:
Input $v \in \{0,1\}^{2^n}$
Accept if and only if $\kappa(uv) \le (1 + 1/n^2)\kappa(u)$.
```

Figure 4.5: Statistical Test $T_{\beta,u}$ for $\Gamma_\beta(\mathrm{P})$-Martingales

Following the lead of Theorem 4.5, let us now analyze the complexity of implementing the statistical test $T_{\beta,u}$. For any $w$ and any $k \le |w|$, let $w_k$ denote the $k$-th bit of $w$, let $w_{1\ldots k}$ denote the prefix of $w$ that contains the first $k$ bits of $w$, and when $k = 0$, let $w_{1\ldots k} = \lambda$, the empty string. Taking $\kappa(\lambda) = 1$,

$$\kappa(w) = 1 + \frac{1}{2}\sum_{k=0}^{|w|-1} (-1)^{1+w_{k+1}} \beta(w_{1\ldots k}).$$

If a circuit that computes $T_{\beta,u}$ attempts to compute $\kappa(uv)$ and $\kappa(v)$, it will have to evaluate summations like the above. This doesn't seem possible with a circuit of size polynomial (in $2^n$) and constant depth. The best that one can do appears to be polynomial (in $2^n$) size and constant depth *using* majority gates. However, we don't know of any pseudorandom generator provably secure against such circuits! In other words, we don't seem to be able to follow the strategy used in the proof of Theorem 4.5 to choose a $v$ deterministically from the range of Nisan's generator.

Since the approach of using a pseudorandom generator based on the computability of $T_{\beta,u}$ does not seem to work, our only hope seems to be to analyze *why* $T_{\beta,u}$ would accept a random $v$ with reasonable probability, and then attempt to *mimic* that reason deterministically. To this end, we will rework the proof of the following statement:

$$\Pr_{v \in \{0,1\}^{2^n}}[\kappa(uv) > r\kappa(u)] < \frac{1}{r}.$$

Let us begin by computing the expectation of $\kappa(uv)$ when $u$ is fixed and $v$ is chosen uniformly at random from all 0-1 strings of length $2^n$. Since $\kappa$ is a martingale, this expectation is clearly $\kappa(u)$. However, we know something more about the martingale $\kappa$: the induced betting function $\beta$ depends only on a few bits of the input, and $\kappa$ can be expressed as a sum of $\beta(\cdot)$'s. Let us see where this leads us:

$$\begin{aligned}
\mathrm{Exp}_{v\in\{0,1\}^{2^n}}[\kappa(uv)] &= \mathrm{Exp}_{v\in\{0,1\}^{2^n}}\left[\kappa(u) + \frac{1}{2}\sum_{i=1}^{2^n-1}(-1)^{1+v_{i+1}}\beta(uv_{1\ldots i})\right] \\
&= \kappa(u) + \frac{1}{2}\sum_{i=1}^{2^n-1}\mathrm{Exp}_{v\in\{0,1\}^{2^n}}\left[(-1)^{1+v_{i+1}}\beta(uv_{1\ldots i})\right] \\
&= \kappa(u),
\end{aligned}$$

since the terms $(-1)^{1+v_{i+1}}\beta(uv_{1\ldots i})$ and $(-1)^{1+\widetilde{v_{i+1}}}\beta(uv_{1\ldots i})$ cancel each other. At this point, we may apply Markov's inequality and conclude that $\Pr_{v\in\{0,1\}^{2^n}}[\kappa(uv) > r\kappa(u)] < \frac{1}{r}$.

Have we gained any new insight into how to deterministically produce a $v$ that passes the statistical test $T_{\beta,u}$? Let us recall that our goal is to construct a small sample space $D$ of $2^n$-bit strings from which we can sample $v$ and still achieve the same probability of $1/n^2$ in passing the test $T_{\beta,u}$. The foregoing analysis indicates that we only need that for any $i$, $\mathrm{Exp}_{v\in D}[(-1)^{1+v_{i+1}}\beta(uv_{1\ldots i})] = 0$, and we know that the quantity $(-1)^{1+v_{i+1}}\beta(uv_{1\ldots i})$ depends only on $\log^c |uv| = O(n^c)$ bits of $v$. Thus, if we take $D$ to be a sample space that provides $2^n$ 0-1 random variables that are $O(n^c)$-wise independent, then we would've solved the problem!

In more detail: If we fix any $u$ and choose $v$ from a sample space $D$ of $2^n$ 0-1 random variables that are $O(n^c)$-wise independent, then for any $u$, $\mathrm{Exp}_{v\in D}[\kappa(uv)] = \kappa(u)$, and by Markov's inequality, the probability that $\kappa(uv) \le (1+1/n^2)\kappa(u)$ is at least $1/(n^2+1)$. In particular, there must exist at least one $v \in D$ such that this event happens. Now we can appeal to the algorithm of Figure 4.3 to produce a language $L$ on which $\kappa$ does not become infinity in the limit.

What is the complexity of the language $L$ thus constructed? That, of course, depends on the choice of the sample space $D$. Let $N = 2^n$, and let $k = \log^c(2^{n+1})$. The most well-known sample space that provides $N$-bit strings whose bits are $k$-wise independent is due to Alon, Babai, and Itai [ABI86]. However, it turns out that if we use this sample space, we can only prove that $L \in \mathrm{NC}^1$ (which would still be a meaningful result, but not the best possible). Once again, we turn to the marvelous generator of Nisan's for rescue: recall that we showed in Fact 4.2 that if we construct Nisan's generator with parameters $N$ and $M$, the resulting sample space provides $N$ bits that are $(M/\log N)$-wise independent. Thus we set $M/\log N = k$, so that $M = k\log N = O(n^{c+1})$, and the seed size for Nisan's generator is approximately $M^2 = O(n^{2c+2})$, which is polynomial in $n$ for any fixed $c$. Moreover, we already know (from Fact 4.3) that with these settings, the language $L$ can be placed into $\mathrm{AC}^0_4[\oplus]$. $\square$

## 4.7 In Retrospect

In this chapter, we explored various notions of measure zero at the polynomial-time level, and considered the measure of the low-level (nonuniform) circuit complexity classes AC$^0$ and AC$^0[\oplus]$. We make some quick closing remarks on various aspects of the work reported in this chapter.

First, we briefly analyze the various notions of measure on P. We considered four notions of measure zero in P that are defined based on whether the martingale outputs the "current capital" or the "next bet," and on whether the capital function goes to infinity in the limit or becomes merely unbounded (lim sup infinity). Allender and Strauss [AS95] have shown that for martingales that compute the $\kappa$ function, success via lim sup is strictly weaker than success via limit. Our result in Section 4.4 placed a limit on the power of $\Gamma_\kappa(\mathrm{P})$-martingales that succeed via lim sup: we showed that no such martingale can cover all of nonuniform $\mathrm{AC}^0_4[\oplus]$. Notice that such martingales can "play" an exponential number of disjoint strategies (or "shares") at each stage $n$; yet they are forced (by the requirement to compute $\kappa$) to restrict the strategies to contiguous segments of $n^{O(1)}$ bits. The notion of a $\Gamma_\beta(\mathrm{P})$-martingale relaxes this, and allows the martingales to play exponential number of disjoint strategies at each stage $n$, with the only restriction that the positions that define each strategy must be computable in time $\log^{O(1)}|w|$ on an input $w$. This difference seems to be quite critical, for we showed in Section 4.5 that all of AC$^0$, and in fact, the much larger class $\widehat{\mathrm{AC}^0_d}$ of languages accepted by constant-depth, nearly exponential-size circuits has $\Gamma_\beta(\mathrm{P})$-measure zero. In Section 4.6, we showed that the above result is nearly optimal, since nonuniform $\mathrm{AC}^0_4[\oplus]$ does not have $\Gamma_\beta(\mathrm{P})$-measure zero.

An interesting open question that remains is the power of $\Gamma_\beta(\mathrm{P})$-martingales that succeed by lim sup. This author's belief is that they aren't too much more powerful, and in fact, it should be possible to show that $\mathrm{AC}^0[\oplus]$ does not have measure zero under this notion either. In fact, this author believes that the two notions of measure that arise from $\Gamma_\beta(\mathrm{P})$-martingales by defining success by limit and by lim sup are the same.

Perhaps the more intriguing story in this chapter is the versatility of Nisan's pseudorandom generator secure against constant-depth circuits. The essence of the ideas used in Sections 4.4 and 4.5 are in the lower bound techniques and results against constant-depth circuits. However, the lower bound techniques, in particular, the Switching Lemma type arguments turn out to be insufficient in proving these results. It is through the use of Nisan's pseudorandom generator that we could take advantage of the lower bounds; thus the pseudorandom generator acts as a very useful catalyst in harnessing the full power of the strong lower bounds. In the result of Section 4.6, this generator turned out to play the crucial role again, but not because it is secure against constant-depth circuits, but for a different reason: the quasirandom property of $k$-wise independence that is inherent in the sample space produced by the generator. Incidentally, the relationship between $k$-wise independence and pseudorandomness against constant-depth circuits is a subject of active study in complexity theory, and there is a prominent outstanding conjecture [LN90] relating the two notions. Partial progress on this question has been reported by Sitharam [Sit94].

# Bibliography

[ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost $k$-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–303, 1992. A preliminary version appeared in FOCS 1991.

[AKS87] M. Ajtai, J. Komlos, and E. Szemeredi. Deterministic simulation in Logspace. In *Proc. 19th Annual ACM Symposium on the Theory of Computing*, 1987.

[All89] E. Allender. Some consequences of the existence of pseudorandom generators. *J. Comp. Sys. Sci.*, 39:101–124, 1989.

[ALM$^+$92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[Aro95] S. Arora. Reductions, codes, PCPs, and inapproximability. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 404–413, 1995.

[AS92a] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 1992. With an appendix by P. Erdős.

[AS92b] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[AS94a] E. Allender and M. Strauss. Measure on small complexity classes, with applications for BPP. In *Proc. 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 807–818, 1994.

[AS94b] E. Allender and M. Strauss. Toward a measure for P. TR 94-14, DIMACS, 1994.

[AS95] E. Allender and M. Strauss. Measure on P: Robustness of the notion. In *Proc. 20th International Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 129–138. Springer-Verlag, 1995.

[BCG$^+$92] S. Buss, S. Cook, A. Gupta, , and V. Ramachandran. An optimal parallel algorithm for formula evaluation. *SIAM Journal on Computing*, 21:755–780, 1992.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Pseudorandom functions revisited: The cascade construction, 1996. To appear in FOCS 1996.

[BFKL94] A. Blum, M. Furst, M. Kearns, and R. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology—CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer-Verlag, 1994.

[BGS75] T. Baker, J. Gill, and R. Solovay. relativizations of the P=NP? question. *SIAM Journal on Computing*, 4:431–442, 1975.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–321, 1977. A preliminary version appeared in STOC 1976.

[BH89] R. Boppana and R. Hirschfeld. Pseudorandom generators and complexity classes. In S. Micali, editor, *Advances in Computing Research*, volume 5, pages 1–26. JAI Press Inc, 1989.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[Bus87] S. Buss. The boolean formula value problem is in ALOGTIME. In *Proc. 19th Annual ACM Symposium on the Theory of Computing*, pages 123–131, 1987.

[BvzGH82] A. Borodin, J. von zur Gathen, and J. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52:241–256, 1982.

[Cai86] J. Cai. Almost optimal lower bounds for small-depth circuits. In *Proc. 18th Annual ACM Symposium on the Theory of Computing*, pages 21–29, 1986.

[Cai89] J. Cai. With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy. *J. Comp. Sys. Sci.*, 38:68–85, 1989. A preliminary version appeared in STOC 1986.

[Chi85] A.L. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *Proc. 5th International Conference on Fundamentals of Computation Theory*, Lecture Notes in Computer Science, pages 63–69. Springer-Verlag, 1985.

[CLL$^+$95] J. Cai, R. Lipton, L. Longpré, M. Ogihara, K. Regan, and D. Sivakumar. Communication complexity of key agreement on limited ranges. In *Proceedings of the Annual Symposium on Theoretical Aspects of Computer Science*, number 834 in Lecture Notes in Computer Science, pages 56–65. Springer-Verlag, 1995.

[CNS96] J. Cai, A. Naik, and D. Sivakumar. On the existence of hard sparse sets under weak reductions. In *Proc. 13th Annual Symposium on Theoretical Aspects of Computer Science*, pages 307–318, 1996.

[Coo85] S. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[CS95a] J. Cai and D. Sivakumar. The resolution of a Hartmanis conjecture. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 362–373, 1995.

[CS95b] J. Cai and D. Sivakumar. Resolution of Hartmanis' conjecture for NL-hard sparse sets, 1995. Manuscript; Revision of UBCS Technical Report 95-40, Department of Computer Science, University at Buffalo, 1995.

[CSS96] J. Cai, D. Sivakumar, and M. Strauss. Constant-depth circuits and the lutz hypothesis, 1996. Manuscript.

[CW79] J. Carter and M. Wegman. Universal classes of hash functions. *J. Comp. Sys. Sci.*, 18:143–154, 1979.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 14–19, 1989.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Info. Thy.*, 22:644–655, 1976.

[For79] S. Fortune. A note on sparse complete sets. *SIAM Journal on Computing*, 8(3):431–433, 1979.

[FSS81] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. In *Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science*, pages 260–270, 1981.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Sys. Thy.*, 17:13–27, 1984.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33:792–807, 1986.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., New York, 1979.

[GL89] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. 21st Annual ACM Symposium on the Theory of Computing*, pages 25–32, 1989.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18:186–208, 1989.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or All languages in NP have zero-knowledge proof systems. *J. ACM*, 38:691–729, 1991.

[Gol95] O. Goldreich. *Foundations of Cryptography: Fragments of a Book*. Electronic Colloquium on Computational Complexity (ECCC), 1995. Available via FTP at `ftp.eccc.uni-trier.de` in the directory `/pub/eccc/books/Oded/`.

[GS89] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In S. Micali, editor, *Advances in Computing Research*, volume 5, pages 73–90. JAI Press Inc, 1989.

[Har78] J. Hartmanis. On log-tape isomorphisms of complete sets. *Theoretical Computer Science*, 7(3):273–286, 1978.

[Hås86] J. Håstad. Almost optimal lower bounds for small-depth circuits. In *Proc. 18th Annual ACM Symposium on the Theory of Computing*, pages 6–20, 1986.

[Hås89] J. Håstad. Almost optimal lower bounds for small-depth circuits. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, CT, USA, 1989. A preliminary version appeared in STOC 1986.

[Hås90] J. Håstad. Pseudorandom generation under uniform assumptions. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 395–404, 1990.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71:231–243, 1986.

[HILL91] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. Technical Report 91–68, ICSI, Berkeley, 1991.

[HOT94] L. Hemachandra, M. Ogiwara, and S. Toda. Space-efficient recognition of sparse self-reducible languages. *Computational Complexity*, 4:262–296, 1994.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117:285–306, 1965.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison–Wesley, Reading, MA, 1979.

[ILL89] R. Impagliazzo, L. Levin, and M. Luby. Pseudo-random generation from one-way functions (extended abstract). In *Proc. 21st Annual ACM Symposium on the Theory of Computing*, pages 12–24, 1989.

[Imm87] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, 1987.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.

[IN89] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. In *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 236–241, 1989.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[JL93] D. Juedes and J. Lutz. The complexity and distribution of hard problems. In *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 177–185, 1993. To appear in SIAM Journal on Computing.

[KL82] R. Karp and R. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982. A preliminary version appeared in STOC 1980.

[Lad75] R. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):18–20, 1975.

[LM94] J. Lutz and E. Mayordomo. Cook versus Karp-Levin: Separating completeness notions if NP is not small. In *Proc. 11th Annual Symposium on Theoretical Aspects of Computer Science*, volume 775 of *Lecture Notes in Computer Science*, pages 415–426. Springer-Verlag, 1994.

[LN86] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge University Press, 1986.

[LN90] N. Linial and N. Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.

[Lut92a] J. Lutz. Almost everywhere high nonuniform complexity. *J. Comp. Sys. Sci.*, 44:220–258, 1992.

[Lut92b] J. Lutz. One-way functions and balanced NP, 1992. Manuscript.

[Lut93] J. Lutz. The quantitative structure of exponential time. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 158–175, 1993.

[Mah82] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *J. Comp. Sys. Sci.*, 25(2):130–143, 1982. A preliminary version appeared in FOCS 1980.

[May92] E. Mayordomo. Almost every set in exponential time is P-bi-immune. In *Proc. 17th International Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 392–400. Springer-Verlag, 1992. To appear in Theoretical Computer Science.

[May94] E. Mayordomo. *Contributions to the Study of Resource-Bounded Measure*. PhD thesis, Universidad Polytécnica de Catalunya, Barcelona, July 1994.

[Mel96] D. Van Melkebeek. Reducing P to a sparse set using a constant number of queries collapses P to L. In *Proc. 11th Annual IEEE Conference on Structure in Complexity Theory*, pages 88–96, 1996.

[Mul87] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987.

[Nis91] N. Nisan. Pseudorandom bits for constant-depth circuits. *Combinatorica*, 11:63–70, 1991.

[Nis92] N. Nisan. *Using hard problems to create pseudorandom generators*. MIT Press, 1992. Revision of author's 1988 U.C. Berkeley thesis for the ACM Distinguished Dissertation series.

[Nis93] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107:135–144, 1993.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal on Computing*, 22:838–856, 1993. A preliminary version appeared in STOC 1990.

[NR95] M. Naor and O. Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 170–181, 1995.

[NRS94] A. Naik, K. Regan, and D. Sivakumar. Quasilinear time complexity theory, 1994. Submitted to *Theoretical Computer Science*. An earlier version appeared in the proceedings of STACS 1994, LNCS 778, pp 97–108.

[NW88] N. Nisan and A. Wigderson. Hardness vs. randomness. In *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–11, 1988.

[NZ93] N. Nisan and D. Zuckerman. More deterministic simulation in Logspace. In *Proc. 25th Annual ACM Symposium on the Theory of Computing*, pages 235–244, 1993.

[Ogi95] M. Ogihara. Sparse hard sets for P yield space-efficient algorithms. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 354–361, 1995.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal on Computing*, 20(3):471–483, 1991. A preliminary version appeared in STOC 1990.

[Oxt80] J. Oxtoby. *Measure and Category*. Springer-Verlag, New York, 1980.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison–Wesley, Reading, MA, 1994.

[Raz94] A. Razborov. Unprovability of lower bounds on circuit size in certain fragments of bounded arithmetic. *Izvestiya of the RAN*, 1994.

[Rom95] S. Roman. *Field Theory*. Springer-Verlag, 1995.

[RR92] D. Ranjan and P. Rohatgi. On randomized reductions to sparse sets. In *Proc. 7th Annual IEEE Conference on Structure in Complexity Theory*, pages 239–242, 1992.

[RR94] A. Razborov and S. Rudich. Natural proofs. In *Proc. 26th Annual ACM Symposium on the Theory of Computing*, pages 204–213, 1994.

[RS95] K. Regan and D. Sivakumar. Improved resource-bounded Borel-Cantelli and stochasticity theorems. UBCS–TR 95-08, Computer Science Dept., University at Buffalo, 1995.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21:120–126, 1978.

[RSC95] K. Regan, D. Sivakumar, and J. Cai. Pseudorandom generators, measure theory, and natural proofs. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 26–35, 1995.

[Sav73] W. Savitch. Maze recognizing automata and nondeterministic tape complexity. *J. Comp. Sys. Sci.*, 7:389–403, 1973.

[Sha81] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. In *Proc. 8th Annual International Conference on Automata, Languages, and Programming*, number 62 in Lecture Notes in Computer Science, pages 544–550. Springer-Verlag, 1981.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proc. 15th Annual ACM Symposium on the Theory of Computing*, pages 330–335, 1983.

[Sip88] M. Sipser. Expanders, randomness or time vs. space. *J. Comp. Sys. Sci.*, 36:379–383, 1988.

[Sit94] M. Sitharam. Pseudorandom generators and learning algorithms for $\mathrm{AC}^0$. In *Proc. 26th Annual ACM Symposium on the Theory of Computing*, pages 478–486, 1994.

[Sto77] L. Stockmeyer. The polynomial time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Str96] M. Strauss. Measure on P: strength of the notion. TR 95-26, Department of Computer Science, Iowa State University, January 1996.

[Sze87] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Bull. of the EATCS*, 33:96–100, 1987.

[vL91] J.H. van Lint. *Introduction to Coding Theory*. Springer-Verlag, 1991.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.

[Wig94] A. Wigderson. $\mathrm{NL/poly} \subseteq \oplus\mathrm{L/poly}$. In *Proc. 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 59–62, 1994.

[Yao82] A. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.