The complexity and distribution of computationally useful problems

David W. Juedes
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd
Part of the Computer Sciences Commons, and the Mathematics Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/10613

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600
The complexity and distribution of computationally useful problems

Juedes, David W., Ph.D.
Iowa State University, 1994
The complexity and distribution of computationally useful problems

by

David W. Juedes

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Department: Computer Science
Major: Computer Science

Approved:
Signature was redacted for privacy.

In Charge of Major Work
Signature was redacted for privacy.

For the Major Department
Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1994
# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** .............................................. v

**CHAPTER 1. INTRODUCTION** ......................................... 1

  1.1 Complexity in Nature ........................................... 2
  1.2 Accessible Information and Computational Usefulness .......... 5
  1.3 Overview ...................................................... 8

**CHAPTER 2. FUNDAMENTAL CONCEPTS FROM COMPLEXITY THEORY** ................................. 13

  2.1 Basic Notation and Terminology ................................ 13
  2.2 Models of Computation and Complexity Classes ................. 15
  2.3 Reducibilities and Hardness .................................. 18
  2.4 Measure and Category ......................................... 20
    2.4.1 Resource-bounded Measure ................................ 21
    2.4.2 Baire Category ........................................... 29
  2.5 Algorithmic Information and Randomness ........................ 33
    2.5.1 Resource-bounded Kolmogorov Complexity ................ 34
    2.5.2 Algorithmic Information Theory ............................ 35
    2.5.3 Algorithmic Randomness .................................... 46
LIST OF FIGURES

Figure 1.1: The dependency relation among the chapters. 12
Figure 2.1: A self-delimiting Turing machine. 36
Figure 2.2: The Turing Machine $M$ used in the proof of Lemma 2.10. 43
Figure 3.1: The Turing machine $M'$ used in the proof of Lemma 3.5. 57
Figure 3.2: A classification of binary sequences. RAND has measure 1, while wkDEEP $-$ strDEEP is comeager. 66
Figure 5.1: An algorithm that computes $\chi_{A^m}$. 97
Figure 5.2: The algorithm for $\hat{M}_A$ in the proof of Theorem 5.13. 108
Figure 5.3: The machine $M$. 111
Figure 5.4: The construction of an $\leq_{m}^{P}$-complete language with high KS a.e. 115
ACKNOWLEDGEMENTS

First and foremost I wish to thank my advisor, Jack Lutz, for his constant patience and support, and his many timely suggestions. I am forever grateful to him for introducing me to structural complexity theory and for always being a good friend. I also wish to thank my many friends and colleagues at Iowa State University who made my years there so interesting and enjoyable, especially Amy Ackerberg, Josef Breutzmann, Ann Grimm, Nicole Lang, James Lathrop, Kesig Lee, David Martin, Susan Mayer (Stapleton), Sunil Nair, Linda Null, Hitendra Patel, Susan Wahls, Timothy Wahls, and Jihoon Yang. I also wish to thank my Ph.D. committee, Daniel Ashlock, David Fernández-Baca, Jonathan Smith, and Giora Slutzki, for their input and suggestions on my work. Finally, I wish to thank my parents, brother, and sisters for their support during my long years at school. I will never forget the life and love that they have given me.

I gratefully acknowledge Applied Communications, Inc. and the Arthur Collins Foundation for providing support for my graduate education through their generous scholarships. My thesis research was also supported in part by National Science Foundation grants CCR-8809238 and CCR-9157382, with matching funds from Rockwell International, Microware Systems Corporation, and Amoco Foundation.

Much of the work contained here was performed jointly with Jack Lutz and
James Lathrop. I am eternally grateful for their contributions to this work and for their permission to include this work here.
CHAPTER 1. INTRODUCTION

What do a sequence of DNA, an encyclopedia, and the UNIX operating system have in common? All three of these objects contain large amounts of useful information, organized in such a fashion as to be readily available to biological, intellectual, or computational processes. Similarly, the solutions of certain natural decision problems, such as the halting problem or the boolean satisfiability problem, contain large amounts of useful information about computation that is readily available to efficient computational processes. Here we search for the common thread among these objects.

This dissertation investigates classes of problems that are computationally useful. In particular, we investigate the complexity and distribution of problems that contain large amounts of useful information about computation. The main results of this dissertation are of the following three general types.

1. Useful problems contain highly organized information.

2. Very useful problems are so highly organized that they are unusually simple and hence rare.

3. Useful problems are, as a whole, not rare and thus are not necessarily simple.

Our main result of type (1) has its roots in the search for a measure that accu-
rately reflects the complexity of natural objects. Our results of type (2) and (3) are motivated by the study of intractability in structural complexity theory. We first examine our main result of type (1). We begin with a brief digression on the complexity of objects in nature.

1.1 Complexity in Nature

The spectrum of complexity in nature ranges from the exceedingly simple to the exceedingly complex. On the low end of the spectrum lie very simple objects: perfect crystals, pure liquids, gases, etc. On the high end of the spectrum lie very complex objects: DNA, human beings, societies, etc. In most cases, objects at the high end of the complexity spectrum are the result of the action, over time, of some complex adaptive system. The best example of this phenomenon is the amazing variety of life on earth. Each organism on this planet, including ourselves, is the result of the action of evolution over a span of billions of years. In the light of these phenomena, we are left wonder about the origin, nature, and driving principles of this vague notion of natural complexity.

Recently [82, 90], there has been a concerted effort among scientists from various disciplines to uncover the nature and origins of complexity. Part of that investigation has concentrated on the complexity of the binary representations of physical and mathematical objects. This part of the investigation has its origins in algorithmic information theory.

Algorithmic information theory provides a rigorous, quantitative measure of the information content of individual binary strings and binary sequences. Briefly, the algorithmic information content of an individual binary string is the length of the
Algorithmic information theory alone does not satisfy our intuition about the complexity of natural objects. In contrast to the aforementioned spectrum of complexity in nature, the strings occupying the high end of the algorithmic information scale are completely random. Such strings might represent the positions of individual atoms in a volume of gas or some other nearly random phenomenon, but they most likely do not represent objects on the high end of nature's complexity spectrum. Consider the following example.

Let $\alpha_1$ and $\alpha_2$ be two strings over the alphabet $\{0, 1\}$. Let $\alpha_1$ be the encoding of one million bits of genetic information from one of the reader’s chromosomes and let $\alpha_2$ be a string of equal length that is chosen at random. (Here we assume that each base-pair in a DNA molecule is encoded by two bits, with 00 representing adenine, 01 representing guanine, 10 representing cytosine, and 11 representing thiamine.) Since the human genome contains a number of repetitive sequences\(^1\), the string $\alpha_1$ can be substantially compressed. Under the conservative assumption that 15% of the human genome consists of repetitive sequences of base-pairs, we should be able to write a complete description of $\alpha_1$ using less than 900,000 bits. Thus the algorithmic

---

\(^1\)One family of repetitive sequences comprises approximately 3% of the human genome [32, p.102-105]. It is possible that as much as 20-30% of the human genome consists of repetitive sequences.
information content of \( \alpha_1 \) is at most 900,000 bits. On the other hand, it is very unlikely that \( \alpha_2 \) can be compressed this much. Since there are \( 2^{1,000,000} \) possible \( \alpha_2 \)'s and at most \( 2^{900,001} \) possible unique binary descriptions of length \( \leq 900,000 \), the probability that the information content of \( \alpha_2 \) is less than 900,000 bits is 1 in \( 2^{99,999} \).

The above example illustrates the crux of the problem with traditional algorithmic information theory, namely, that it provides no means to measure accessible information or the organization of information in a given string. The string \( \alpha_1 \) contains a large amount of information that is organized in such a way that it is readily accessible to biological processes. Conversely, the string \( \alpha_2 \) probably contains even more information than \( \alpha_1 \), but that information has no organization and is not readily accessible to biological processes.

To remedy the above situation, Bennett [11] extended algorithmic information theory to include a notion of computational depth for binary strings and sequences. Roughly speaking, the computational depth of a string is the number of steps required to produce the string from its shortest complete description. The rationale behind such a measure is that truly intricate strings should take much longer to generate from their compressed versions than either simple or incompressible strings. Bennett’s notion appears to rigorously capture the level of organization in a string.

For infinite binary sequences, Bennett [11] defines notions of strong and weak computational depth. These notions intuitively represent the level of organization in infinite binary sequences. To justify his intuition, Bennett shows that no recursive or algorithmically random sequence can be either strongly or weakly deep. Moreover, Bennett shows that \( K \), the diagonal halting language (see [33], for example), is strongly deep.
The fact that $K$ is strongly deep deserves careful scrutiny. It is well-known that $K$ is efficiently many-one hard for the set RE of all recursively enumerable languages\textsuperscript{2}. Roughly speaking, this means that the membership of any string $x$ in any recursively enumerable language $L$ can be determined by answering a single question about the membership of another string $y$ in $K$, where $y$ can be efficiently computed from $x$. Thus $K$ contains a large amount of accessible information about the recursively enumerable languages. One might even say that $K$ is computationally useful. The natural question to ask is the following. Is $K$ strongly deep because of its organization or is $K$ strongly deep because of the amount of accessible information it contains? In Chapter 3 we answer this question. $K$ is strongly deep precisely because it contains a large amount of accessible information. Let us now examine what we mean by accessible information.

1.2 Accessible Information and Computational Usefulness

Whereas algorithmic information theory quantifies the "amount of information" in a mathematical object and computational depth appears to quantify the "amount of organization" in a mathematical object, we quantify the "amount of accessible information" in a mathematical object by measuring the size of the set of objects that can be efficiently generated when given access to the original object. For infinite binary sequences, our notion of accessible information is completely rigorous. Unfortunately, we know of no analogous notion for finite strings or the physical objects they represent. For this reason, we now leave our discussion of the underpinnings of

\textsuperscript{2}A language $L$ is recursively enumerable if there is a Turing machine that enumerates all the elements of $L$.  

"natural" complexity and delve into the world of structural complexity theory. (We assume at this point that the reader is familiar with the basic notation and terminology of structural complexity theory. This notation can be found in [7, 8] and is reviewed in Chapter 2 for completeness.)

From this point on, the primary mathematical objects that we consider are infinite binary sequences. Note, however, that there is a one-to-one correspondence between infinite binary sequences and languages, i.e., subsets of \( \{0,1\}^* \), and therefore we use these terms interchangeably. Here we consider the "amount of accessible information" in languages. More precisely, we consider the amount of accessible information about certain complexity classes.

A precise notion of the "amount of accessible information" in the above sense requires a measure of the size of a set inside a complexity class. For this purpose, we turn to resource-bounded measure, a resource-bounded generalization of classical Lebesgue measure developed by Lutz [63, 66]. Resource-bounded measure requires a brief explanation.

For a set of languages \( X \) and complexity class \( C \), resource-bounded measure provides a quantity that can be loosely interpreted as the conditional probability that a language is in \( X \) given that it is in \( C \). More precisely, resource-bounded measure defines a set of languages \( X \) to have measure 0 in \( C \) if \( X \cap C \) is a negligible subset of \( C \) as witnessed by a resource-bounded betting strategy. Similarly, the set \( X \) is defined to have measure 1 in \( C \) if \( X^c \cap C \) has measure 0 in \( C \). For notational convenience, we say that \( X \cap C \) is a non-negligible subset of \( C \) if \( X \) does not have measure 0 in \( C \). Furthermore, we say that almost every language in \( C \) is in \( X \) if \( X \) has measure 1 in \( C \). We review these basic definitions and the properties of resource-bounded measure.
in Section 2.4.

Let us now make the notion of accessible information more precise. We associate "accessible information" with information that is accessible to some resource-bounded reducibility. Thus the sum total of the accessible information in a language is precisely the set of languages that are reducible to it, i.e., its lower reducibility span. Appropriately, we measure the "amount of accessible information" in some language by using the resource-bounded measure of its lower reducibility span. (This notion of "accessible information" was proposed by Lutz [60, 61] and subsequently investigated in [37, 38, 40, 65, 67, etc.].)

In general, the above notion of accessible information is too cumbersome to work with directly. We do, however, work with the notion indirectly. We examine languages that contain non-zero amounts of accessible information. These languages are said to be weakly hard. More precisely, we say that a language $H$ is weakly hard for a complexity class $C$ and reducibility $\leq_r$ if the set $\{A \mid A \leq_r H\}$ does not have measure 0 in $C$. This notion of weak hardness was first introduced by Lutz [60, 61] and has the classical notion of hardness [7, 79] as a special case.

Since each weakly hard language contains a large amount of accessible information about some complexity class, we contend that these languages are computationally useful. As evidence to support this contention, consider the following example. Let $H$ be a language that is weakly $\leq_m^P$-hard for $E$. From the definition, the set $P_m(A) = \{A \mid A \leq_m^P H\}$ does not have measure 0 in $E$. In contrast, it is well-known [63] that for any constant $c$ the set $\text{DTIME}(2^{cn})$ has measure 0 in $E$. It follows that for every constant $c \in \mathbb{N}$, there is a language in $E - \text{DTIME}(2^{cn})$ that is $\leq_m^P$-reducible to $H$. Thus $H$ can be used to solve problems that are arbitrarily difficult in $E$. Similar
situations occur in other complexity classes. For this reason, we say that weakly hard languages are computationally useful.

1.3 Overview

The work contained herein is a unified and rigorous examination of many recent results from the ongoing investigation of computationally useful problems. This dissertation contains results compiled from joint work with James Lathrop and Jack Lutz [37, 38, 39, 40], as well as some new results. The primary body of this work consists of a rigorous examination of the measure-theoretic structure of the complexity classes $E = \text{DTIME}(2^{\text{linear}})$, $E_2 = \text{DTIME}(2^{\text{polynomial}})$, $\text{ESPACE} = \text{DSPACE}(2^{\text{linear}})$, $E_2\text{SPACE} = \text{DSPACE}(2^{\text{polynomial}})$, and $\text{REC}$, the set of all recursive languages. Particular attention is paid to languages that are weakly hard in the previously mentioned sense.

In Chapter 3, we examine Bennett’s notion of computational depth. We review and investigate Bennett’s notions of strong and weak computational depth. There we define a language $x$ to be weakly useful if there is a recursive time bound $s$ such that $\text{DTIME}^x(s)$ does not have measure 0 in $\text{REC}$, i.e., if it is weakly hard for $\text{REC}$. The main result of Chapter 3 shows that every weakly useful language is strongly deep. Since $K$ is weakly useful, this resolves our previous question and provides an alternate proof of Bennett’s original result.

Chapter 4 consists of a measure-theoretic investigation of the structure of $E$ and $E_2$. There we exhibit an almost everywhere lower bound on the size of complexity cores for languages in $E$ and $E_2$. Furthermore, we exhibit a tight lower bound on the size of complexity cores for the weakly $\leq_{m}^{p}$-hard languages for $E$ and $E_2$ and,
surprisingly, a tight upper bound on the size of complexity cores for the $\leq^p_m$-hard languages for $E$ and $E_2$. Our tight lower bound extends a result of Orponen and Schöning [77] to show that every weakly $\leq^p_m$-hard language for $E$ contains a dense $\text{DTIME}(2^{n^c})$-complexity core. Our almost everywhere lower bound, in combination with our tight upper bound implies the main result of this chapter, namely, that the set of $\leq^p_m$-hard languages for $E$ or $E_2$ forms a measure 0 subset of $E$ or $E_2$.

The main result of Chapter 4 is, in fact, a special case of a more general phenomenon, namely, that each $\leq^p_m$-degree has measure 0 in $E$ or $E_2$. This implies that the sets of $\leq^p_m$-complete languages for $NP$, PSPACE, and many other classes of interest form measure 0 subsets of $E$ or $E_2$.

Similarly, the main body of Chapter 5 consists of a measure theoretic investigation of the nonuniform structure of ESPACE and E₂SPACE. There we establish almost everywhere lower bounds on the space-bounded Kolmogorov complexity and size of nonuniform complexity cores for languages in ESPACE and E₂SPACE. Furthermore, we exhibit tight lower bounds on the complexity of weakly $\leq^{p/poly}_m$-hard languages and tight upper bounds on the complexity of $\leq^{p/poly}_m$-hard languages for ESPACE and E₂SPACE. Our lower bounds extend the work of Lutz [63] and Huynh [34, 36]. The almost everywhere lower bounds, in combination with the tight upper bounds, imply that the set of $\leq^{p/poly}_m$-hard languages for ESPACE or E₂SPACE forms a measure 0 subset of ESPACE or E₂SPACE. As in Chapter 4, this result is a special case of the more general phenomenon that each $\leq^{p/poly}_m$-degree has measure 0 in ESPACE or E₂SPACE.

The results of Chapter 5 immediately imply that $P/Poly$, the set of all languages with polynomial-size circuits, has measure 0 in ESPACE. This result was originally
Chapter 6 examines the distribution of weakly hard problems in $E$, $E_2$, ESPACE, and $E_2$SPACE. For some time, the major open question surrounding weak hardness was whether or not weak hardness was actually a more general notion than hardness. In [65], Lutz resolved this question by establishing the existence of languages that are weakly $\leq^P_m$-hard for $E$, but not $\leq^P_m$-hard for $E$. Here we extend this work in a strong, measure-theoretic sense. In Chapter 6, we show that the sets of weakly $\leq^P_m$-hard languages form non-negligible subsets of the complexity classes $E$, $E_2$, ESPACE, and $E_2$SPACE. In contrast, we show in Chapter 4 that the sets of $\leq^P_m$-hard languages form a measure 0 subset of the complexity classes $E$, $E_2$, ESPACE, and $E_2$SPACE. Thus the sets of languages that are weakly $\leq^P_m$-hard but not $\leq^P_m$-hard for the classes $E$, $E_2$, ESPACE, and $E_2$SPACE do not have measure 0 in their respective classes. These results, in combination with the main result of Chapter 5, imply the existence of languages that are weakly $\leq^P_m$-hard, but not $\leq^{p/poly}_m$-hard, for ESPACE. It is not known whether there are languages that are $\leq^{p/poly}_m$-hard, but not weakly $\leq^P_m$-hard, for ESPACE.

In Chapter 7, we conclude this dissertation by providing an overview of our results, posing several interesting open problems, and musing some more about "natural complexity."

Chapters 2 reviews the necessary background material for Chapters 3-6. Section 2.1 presents the basic notation and terminology that is used throughout this dissertation. For completeness, Section 2.2 reviews the basic machine models and complexity classes. Section 2.3 reviews reducibilities and the notions of hardness and completeness. Section 2.4 provides a brief overview of the necessary definitions.
and important properties of resource-bounded measure, Lebesgue measure and Baire category. Section 2.5 presents an overview of algorithmic information theory and the related notions of algorithmic randomness and resource-bounded Kolmogorov complexity.

Finally, we note that this dissertation is structured so that the interested reader may examine individual chapters without reading all of the preceding material. There is, however, some interdependence among the chapters. This interdependence is depicted in Figure 1.1.
Figure 1.1: The dependency relation among the chapters.
CHAPTER 2. FUNDAMENTAL CONCEPTS FROM COMPLEXITY THEORY

This dissertation employs notation, terminology, and concepts from a variety of areas, including structural complexity theory, resource-bounded measure theory, and algorithmic information theory. This chapter provides a review of the relevant material from each of these areas. Section 2.1 reviews the basic notation and terminology that is used in this dissertation. Sections 2.2 and 2.3 review fundamental concepts from structural complexity theory. Section 2.2 reviews the fundamental models of computation and presents some basic complexity classes. Section 2.3 reviews the classical notions of reducibility, completeness, and hardness. Section 2.4 presents fundamental definitions and theorems from Lebesgue measure, resource-bounded measure, and Baire category. Finally, section 2.5 presents fundamental definitions and theorems of algorithmic information theory. Here we follow most of the accepted conventions in the field, such as those found in the books by Balcazar, Diaz, and Gabarró [7, 8], Hopcroft and Ullman [33], and Li and Vitányi [57].

2.1 Basic Notation and Terminology

To begin with, the symbols $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{Z}^+$ denote the set of natural numbers, the set of integers, and the set of positive integers, respectively. We frequently use
conditions or properties of natural numbers. Let $\phi(n)$ be one such property of the natural numbers. Then, we say that $\phi(n)$ holds almost everywhere (a.e.) if $\phi(n)$ is true for all but finitely many $n \in \mathbb{N}$. Similarly, we say that $\phi(n)$ holds infinitely often (i.o.) if $\phi(n)$ is true for infinitely many $n \in \mathbb{N}$. Here we write $[\phi]$ for the Boolean value of a condition $\phi$. That is, $[\phi] = 1$ if $\phi$ is true, 0 if $\phi$ is false.

Our basic mathematical objects are strings, sequences, languages, functions, and classes. Strings are finite sequences of characters over the binary alphabet $\{0,1\}$. Here we write $\{0,1\}^*$ for the set of all strings. "Sequences" are infinite binary sequences. We write $\{0,1\}^\omega$ for the set of all such sequences. Languages are sets of strings. Functions usually map $\{0,1\}^*$ into $\{0,1\}^*$. A class is either a set of languages or a set of functions.

The sequence of strings over $\{0,1\}$, $s_0 = \lambda, s_1 = 0, s_2 = 1, s_3 = 00, \ldots$, is referred to as the standard enumeration of $\{0,1\}^*$. If $x$ is either a string or a sequence, then we write $x[i..j]$ for the string containing the $i$th through $j$th bits of $x$ numbering from zero. For example, if $x$ is 00101101, then the string $x[2..5]$ is 1011. Moreover, we write $x[n]$ for the $n$th bit of $x$. Finally, if $x \in \{0,1\}^*$ is a string, we write $|x|$ for the length of $x$.

If $A \subseteq \{0,1\}^*$ is a language, then we write $A^c$, $A_{\leq n}$, and $A_{= n}$ for $\{0,1\}^* - A$, $A \cap \{0,1\}^{\leq n}$, and $A \cap \{0,1\}^{= n}$ respectively. As mentioned previously, there is a one-to-one correspondence between languages and sequences. That is, we associate each language $A \subseteq \{0,1\}^*$ with its characteristic sequence $\chi_A \in \{0,1\}^\omega$ defined by

$$\chi_A[n] = [s_n \in A].$$

Similarly, we associate the finite sets $A_{\leq n}$ and $A_{= n}$ with their characteristic strings $\chi_{A_{\leq n}} = \chi_A[0..N-1]$ and $\chi_{A_{= n}} = \chi_A[2^n - 1..N-1]$, where $N = |\{0,1\}^{\leq n}| = 2^{n+1} - 1$. 
Unless otherwise noted, we use a standard string pairing function \(\langle x, y \rangle : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*\) defined by \(\langle x, y \rangle = bd(x)01y\), where \(bd(x)\) is \(x\) with each bit doubled (e.g., \(bd(1101) = 11110011\)). Note that \(|\langle x, y \rangle| = 2|x| + |y| + 2\) for all \(x, y \in \{0,1\}^*\). For each \(g : \{0,1\}^* \rightarrow \{0,1\}^*\) and \(k \in \mathbb{N}\), we also define the function \(g_k : \{0,1\}^* \rightarrow \{0,1\}^*\) by \(g_k(x) = g(0^k, x)\) for all \(x \in \{0,1\}^*\).

We denote the cardinality of a finite set \(A\) by \(|A|\). A language \(D\) is dense if there exists some constant \(c > 0\) such that \(|D_{\leq n}| > 2^c n^c\) a.e. A language \(S\) is sparse if there exists a polynomial \(p\) such that \(|S_{\leq n}| < p(n)\) a.e.. A language \(S\) is co-sparse if \(S^c\) is sparse.

### 2.2 Models of Computation and Complexity Classes

Here we examine both uniform and nonuniform models of computation. Our model of uniform computation is based on the standard Turing machine model as found in [7, 33, etc.]. All of our machines are either deterministic multitape Turing machines or deterministic multitape oracle Turing machines. Most of our machines take a single input, but a few of our machines take inputs of the form \((x, n)\), where \(x \in \{0,1\}^*\) and \(n \in \mathbb{N}\). These machines are assumed to have two input tapes, one for \(x\) and the other for the binary representation \(\beta(n) \in \{0,1\}^*\) of \(n\).

The language accepted by a machine \(M\) is denoted by \(L(M)\). Similarly, \(L(M^A)\) denotes the language accepted by an oracle machine \(M\) using oracle \(A\). The functions \(\text{time}_M(x)\) and \(\text{space}_M(x)\) represent the number of steps and tape cells, respectively, that the machine \(M\) uses on input \(x\). Using this notation, the standard time- and space-bounded complexity classes are defined as follows. If \(t : \mathbb{N} \rightarrow \mathbb{N}\) is a time-bound, then \(\text{DTIME}(t(n))\) is the set of languages \(A\) for which there exist a machine
$M$ and constant $c$ such that $L(M) = A$ and $\text{time}_M(x) \leq c \cdot t(|x|) + c$ for every $x \in \{0,1\}^*$. The classes $\text{DSPACE}(s(n))$, $\text{DTIME}^4(t(n))$, and $\text{DSPACE}^4(t(n))$ are defined similarly.

In this dissertation we examine a variety of uniform complexity classes. In Chapter 3, we primarily examine $\text{REC}$, the complexity class containing all the recursive languages. Briefly, a language $L$ is in $\text{REC}$ if there is a machine $M$ that satisfies $L = L(M)$ and halts on all inputs. In Chapters 4 and 6, we examine the exponential time classes

$$
E = \bigcup_{c=0}^{\infty} \text{DTIME}(2^{cn})
$$

and

$$
E_2 = \bigcup_{c=0}^{\infty} \text{DTIME}(2^{c2^n}).
$$

In Chapter 5, we examine the exponential space classes

$$
\text{ESPACE} = \bigcup_{c=0}^{\infty} \text{DSPACE}(2^{cn})
$$

and

$$
\text{ESPACE}_2 = \bigcup_{c=0}^{\infty} \text{DSPACE}(2^{c2^n}).
$$

We also mention but do not directly examine a variety of other classes, such as $P$, $\text{BPP}$, $\text{NP}$, and $\text{PSPACE}$. These classes have completely standard definitions which we do not repeat here. (See [7], for example.)

Our model of nonuniform computation is in terms of machines that take advice. An advice function is a function $h : \mathbb{N} \to \{0,1\}^*$. If $M$ is a machine and $h$ is an advice function, then language accepted by the machine/advice pair $M/h$ is the set $L(M/h) = \{x \in \{0,1\}^* | \langle x, h(|x|) \rangle \in L(M)\}$. We use the functions $\text{time}_{M/h}(x)$
and $\text{space}_{M/h}(x)$ to denote the number of steps and tape cells, respectively that the machine $M$ takes on input $(x, h(|x|))$.

We are most interested in machines that take polynomially bounded advice. (We say that an advice function $h : \mathbb{N} \to \{0,1\}^*$ is \textit{polynomially bounded} if there exists a polynomial $p$ such that $|h(n)| \leq p(n)$ for every $n \in \mathbb{N}$.) We define the basic nonuniform complexity classes as follows. If $t : \mathbb{N} \to \mathbb{N}$ is a time-bound, then $\text{DTIME}(t(n))/\text{Poly}$ is the set of languages $A$ for which there exist a machine $M$, constant $c$, and polynomially bounded advice function $h$ such that $L(M/h) = A$ and $\text{time}_{M/h}(x) \leq c \cdot t(|x|) + c$ for every $x \in \{0,1\}^*$. The classes $\text{DSPACE}(s(n))/\text{Poly}$, $\text{DTIME}^A(t(n))/\text{Poly}$, and $\text{DSPACE}^A(t(n))/\text{Poly}$ are defined similarly. The class $\text{P/Poly}$ is defined as the union over all polynomials $p$ of the $\text{DTIME}(p(n))/\text{Poly}$ classes. It is well-known [41] that $\text{P/Poly}$ consists exactly of those languages that are computed by polynomial-size Boolean circuits.

Finally, we use a variety of function classes. Each of our function classes is defined by the same set of resources as one of our language classes. Following standard usage, we denote each function class by appending an $F$ to the name of the corresponding language class. Although we do not define each class explicitly, their definitions are clear. For example, the class $\text{DSPACE}(s(n))/\text{Poly}F$ is the set of functions $f : \{0,1\}^* \to \{0,1\}^*$ for which there exist a machine $M$, constant $c$, and polynomially bounded advice function $h$ such that $M/h(x) = f(x)$ and $\text{time}_{M/h}(x) \leq c \cdot t(|x|) + c$ for every $x \in \{0,1\}^*$.
2.3 Reducibilities and Hardness

We now review the classical notions of reducibility, completeness, and hardness. These notions have completely standard definitions which can be found in [7, 33, etc.]. We present the notion of reducibility first.

Intuitively, a reducibility is a method of deciding one language when given information about another language. Here we use resource-bounded versions of Turing, many-one, and truth-table reducibilities. Briefly, a language $A$ is Turing reducible to a language $B$ if there is an oracle Turing machine $M$ such that $A = L(M^B)$. Let $C$ be a complexity class and let $A$ and $B$ be languages. Then we say that $A$ is $C$-Turing reducible to $B$ ("$A \leq_T^C B$") if $A \in C^B$. For example, $A$ is polynomial-time Turing reducible to $B$ ("$A \leq_T^P B$") if $A \in P^B$.

A language $A$ is many-one reducible to a language $B$ if there is a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that $x \in A \iff f(x) \in B$ for every $x \in \{0,1\}^*$. In this case, we say that $A$ is many-one reducible to $B$ via $f$. Let $\mathcal{F}$ be a function class and let $A$ and $B$ be languages. Then $A$ is $\mathcal{F}$ many-one reducible to $B$ if there exists a function $f \in \mathcal{F}$ such that $x \in A \iff f(x) \in B$. For example, $A$ is polynomial-time many-one reducible ("$\leq^P_m$-reducible") to $B$ if there is a function $f \in \text{PF}$ such that $x \in A \iff f(x) \in B$ for every $x \in \{0,1\}^*$.

A language $A$ is truth-table reducible to a language $B$, and we write $A \leq_{tt} B$, if there is a computable function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that

$$f(x) = \langle \langle x_1, x_2, \ldots, x_k \rangle, \alpha \rangle,$$

where $\langle x_1, x_2, \ldots, x_k \rangle$ is a finite set and $\alpha$ is a boolean formula, and $x \in A \iff \alpha(\llbracket x_1 \in B \rrbracket \llbracket x_2 \in B \rrbracket \ldots \llbracket x_k \in B \rrbracket)$.
While the above definition is standard and fully captures the spirit of the “truth-table,” this is not the definition we use here. For convenience, we use the following equivalent definition. A language $A$ is truth-table reducible to a language $B$ if there is a recursive time bound $s : \mathbb{N} \to \mathbb{N}$ such that $A \leq^{\text{DTIME}(s)}_{T} B$.

In the following chapters we primarily use the polynomial-time many-one ($\leq^p_m$), P/Poly many-one ($\leq^{p/poly}_m$), P/Poly Turing ($\leq^{p/poly}_T$), and the truth-table ($\leq_{tt}$) reducibilities. There we find it useful to work with either the lower or upper reducibility span of a given language $L$. Briefly, the lower reducibility span of $L$ is set of languages that are reducible to $L$. We write $P_m(L) = \{A \mid A \leq^p_m L\}$, $P/Poly_m(L) = \{A \mid A \leq^{p/poly}_m L\}$, $P/Poly_T(L) = \{A \mid A \leq^{p/poly}_T L\}$, and $REC_{tt}(L) = \{A \mid A \leq_{tt} L\}$, for the lower reducibility spans of $L$ with respect to the $\leq^p_m$, $\leq^{p/poly}_m$, $\leq^{p/poly}_T$, and $\leq_{tt}$ reducibilities, respectively. Similarly, the upper reducibility span of $L$ is set of languages to which $L$ is reducible. We write $P^{-1}_m(L) = \{A \mid L \leq^p_m A\}$, $P/Poly^{-1}_m(L) = \{A \mid L \leq^{p/poly}_m A\}$, $P/Poly^{-1}_T(L) = \{A \mid L \leq^{p/poly}_T A\}$, and $REC^{-1}_{tt}(L) = \{A \mid L \leq_{tt} A\}$, for the upper reducibility spans of $L$ with respect to the $\leq^p_m$, $\leq^{p/poly}_m$, $\leq^{p/poly}_T$, and $\leq_{tt}$ reducibilities, respectively.

This brings us to the notions of hardness and completeness. If the lower $\leq_r$-reducibility span of a language contains a complexity class $C$, then we say that language is $\leq_r$-hard for $C$. More precisely, a language $H$ is $\leq_r$-hard for a class of languages $C$ if every language $A \in C$ is $\leq_r$-reducible to $H$. Moreover, we say that a language $C$ is $\leq_r$-complete for a class $C$ if $C \in C$ and $C$ is $\leq_r$-hard for $C$. Notice that if $\leq_r = \leq^p_m$ and $C = \text{NP}$, then this is the usual notion of NP-completeness [29].
2.4 Measure and Category

Lebesgue measure, resource-bounded measure, and Baire category each provide well-defined notions of the "size" of infinite sets. These notions are used extensively in Chapters 3 through 6. Here we review the relevant formulations of each of these notions.

We are particularly interested in sets of infinite binary sequences (equivalently, sets of languages) that appear in the context of complexity theory. For the most part, such sets have the property that they are closed under finite variations. (Recall that a set $X$ of languages is closed under finite variations if for every language $A \in X$ and every language $B$, the condition $|A \triangle B| < \infty$ implies that $B \in X$. If $X$ is closed under finite variations, then we say that $X$ is a tail set.) For example, most of the sets from the previous section (e.g., P, P/Poly, NP, E, ESPACE, REC, $P_m(L)$, $P/Poly^{-1}(L)$, etc.) are easily seen to be closed under finite variations. This observation allows us to simplify our discussion of the above notions concerning the "size" of infinite sets.

In each of the above notions, the "size" of a tail set has only three possibilities; it is either small, large, or its size is undefined. This phenomenon corresponds to the well-known Kolmogorov zero-one law for Lebesgue measure [43] and resource-bounded measure [60, 61, 66], and its analog for Baire category [78, p. 85]. For Lebesgue measure, this phenomenon corresponds to the fact that every tail set either has measure 0 (i.e., it is small), measure 1 (i.e., it is large), or is not measurable (i.e., its size is undefined). For resource-bounded measure, this phenomenon corresponds to the fact that every tail set either has measure 0 in some complexity class $C$, measure 1 in $C$, or is not measurable in $C$. For Baire category, this phenomenon corresponds
to the fact that every tail set is either \textit{meager} (i.e., it is small), \textit{co-meager} (i.e., it is large), or it does not have \textit{the property of Baire} (i.e., its size is undefined).

Here we review the above notions of "size" and pay particular attention to the specific formalizations of "small" and "large" sets. Our presentation provides sufficient machinery for the reader to follow the arguments presented herein but is not complete. The interested reader is directed to consult [15, 30, 63, 66, 78] for more general presentations of these notions.

We begin by presenting the fundamentals of resource-bounded measure [63, 66], a resource-bounded generalization of Lebesgue measure. We present Lebesgue measure as a special case of this more general theory.

2.4.1 Resource-bounded Measure

Resource-bounded measure\(^1\) as formulated by Lutz [63, 66] provides a rigorous notion of measure inside various classes, including E, E\(_2\), ESPACE, E\(_2\)SPACE, REC, and \(\{0,1\}^\infty\). For the class \(\{0,1\}^\infty\), resource-bounded measure corresponds exactly to classical Lebesgue measure. For the class REC, resource-bounded measure is closely related to, but not known to be equivalent to, the \textit{effective measure} of Freidzon [27], Melhorn [73], and others. Resource-bounded measure is formulated here both in terms of uniform systems of density functions and in terms of betting strategies (martingales). These formulations of resource-bounded measure are equivalent [63, 66] and share common notational conventions. We present these notational

\(^1\)Resource-bounded measure was originally formulated by Lutz [60, 61] in terms of modulated covering by cylinders. Although all the results in [60, 61] are true, the underlying formulation of resource-bounded measure in [60, 61] was technically flawed. Here we use the emended version of resource-bounded measure as found in [63, 66].
conventions first.

Recall that a string $w \in \{0,1\}^*$ is a prefix of a sequence $y \in \{0,1\}^\infty$ if there is a sequence $z \in \{0,1\}^\infty$ such that $y = wz$. In this case, we write $w \subseteq y$. We denote the set of all sequences whose prefix is $w \in \{0,1\}^*$ by

$$C_w = \{ z \in \{0,1\}^\infty \mid w \subseteq z \}$$

the cylinder specified by $w$.

Both formulations of resource-bounded measure require the approximation of real-valued functions. For this purpose, we use the set

$$D = \{ m \cdot 2^{-n} \mid m, n \in \mathbb{N} \}$$

of nonnegative dyadic rationals. Furthermore, both formulations require precise notions of the computational complexity of such approximations. In order to have an unambiguous criteria for the complexity of these approximations, we assume that all functions $f$ map $\{0,1\}^*$ into $\{0,1\}^*$.

For notational convenience, we often write that a function $f$ maps some set $X$ into $D$, where $X$ is either $\{0,1\}^*$, $\mathbb{N}$, $D$, or some cartesian product of these sets. In this situation, we implicitly associate the function $f : X \rightarrow D$ with a function $\tilde{f} : \{0,1\}^* \rightarrow \{0,1\}^*$ that acts on strings in $\{0,1\}^*$ as encoded elements of $X$ and produces encoded elements of $D$.

**Example:** Let $f$ map $\mathbb{N}^2 \times \{0,1\}^*$ into $D$, let $i, j \in \mathbb{N}$, and let $w \in \{0,1\}^*$. Then we associate $f$ with the function $\tilde{f} : \{0,1\}^* \rightarrow \{0,1\}^*$ that operates as follows. The function $\tilde{f}$ on input $(0^i, 0^j, w)$ acts precisely like $f$ on input $(i, j, w)$, i.e., if $f(i, j, w) = m \cdot 2^{-n}$ then the function $\tilde{f}$ on input $(0^i, 0^j, w)$ produces the string
\langle u, v \rangle$, where \( u, v \) are the binary representations of \( m \) and \( n \). Moreover, \( f \) halts immediately on any input that is not of the form \( \langle 0^i, \langle 0^j, w \rangle \rangle \).

2.4.1.1 Density Systems

Our main formulation of resource-bounded measure is in terms of uniform systems of density functions [63]. We employ this version of resource-bounded measure in Chapters 3 - 5. To begin, consider the following definitions.

**Definition** (Lutz [63]). A *density function* is a function \( d : \{0, 1\}^* \rightarrow [0, \infty) \) that satisfies

\[
d(x) = \frac{d(x0) + d(x1)}{2}
\]

for all \( x \in \{0, 1\}^* \). The *global value* of a density function \( d \) is the value \( d(\lambda) \). The *set covered by* a density function \( d \) is

\[
S[d] = \bigcup_{x \in \{0, 1\}^* \atop d(x) \geq 1} C_x.
\]

A density function \( d \) covers a set \( X \subseteq \{0, 1\}^\infty \) if \( X \subseteq S[d] \).

Notice that the density function \( d \) provides a weighting of the elements of \( \{0, 1\}^* \) such that for any arbitrary \( n \in \mathbb{N} \) at most \( d(\lambda) \cdot 2^n \) of the \( 2^n \) strings in \( \{0, 1\}^n \) have a density of one or more. Hence the density function \( d \) verifies that the statement \("Pr[x \in S[d]] \leq d(\lambda)"\) is true in the standard random experiment, i.e., the experiment in which the bits of \( x \) are chosen by the independent toss of a fair coin.

A single density function \( d \) only provides a means of stating that the set \( X \) contains at most \( d(\lambda) \) of \( \{0, 1\}^\infty \). Since we are primarily interested in “small” sets, we require a means of specifying that a set is arbitrarily small. For this purpose,
we use uniform systems of density functions. Here, an \( n \)-dimensional density system \((n-DS)\) is a function \( d : \mathbb{N}^n \times \{0,1\}^* \rightarrow [0, \infty) \) such that \( d_k \) is a density function for every \( k \in \mathbb{N}^n \). This brings us to the fundamental concept of a null cover.

**Definition** (Lutz [63]). A null cover of a set \( X \) of languages is a 1-DS \( d \) such that, for all \( k \in \mathbb{N} \), \( d_k \) covers \( X \) with global value \( d_k(X) < 2^{-k} \).

It is easy to show [66] that a set \( X \subseteq \{0,1\}^\infty \) has classical Lebesgue measure 0 if and only if there exists a null cover of \( X \). The primary difference between classical Lebesgue measure and resource-bounded measure, as we will see, is that resource-bounded measure adds the requirement that the density systems be uniformly computable in some resource bound.

Since density systems are real-valued, the concept of computing density systems requires approximation. Here we approximate density systems by functions that produce arbitrarily good approximations via dyadic rational numbers. A *computation* of an \( n \)-DS \( d \) is a function \( \hat{d} : \mathbb{N}^{n+1} \times \{0,1\}^* \rightarrow \mathbb{D} \) such that

\[
|\hat{d}_{\vec{k},r}(x) - d_{\vec{k}}(x)| \leq 2^{-r}
\]

for all \( \vec{k} \in \mathbb{N}^n \), \( r \in \mathbb{N} \), and \( x \in \{0,1\}^* \). Thus a computation of an \( n \)-DS \( d \) can be used to approximate \( d \) to any arbitrary precision.

Resource-bounded measure is achieved by adding resource-bounds in the natural way to the above computations. Let \( \Delta \) be a class of functions mapping \( \{0,1\}^* \) into \( \{0,1\}^* \). A \( \Delta \)-computation of an \( n \)-DS \( d \) is a computation \( \hat{d} \) such that \( \hat{d} \in \Delta \). An \( n \)-DS is \( \Delta \)-computable if there exists a \( \Delta \)-computation \( \hat{d} \) of \( d \).

Measure inside of complexity classes is achieved by associating function classes with classes of languages. In [60, 61, 66], Lutz shows that certain function classes
\( \Delta \) characterize classes of languages \( R(\Delta) \). Moreover, Lutz shows that each \( \Delta \) class induces measure structure on the corresponding \( R(\Delta) \) class. Here we use the following \( \Delta \) classes. (Recall from section 2.2 that \( \text{DTIME}(t(n)) \) and \( \text{DSPACE}(s(n)) \) are classes of functions \( f: \{0,1\}^* \to \{0,1\}^* \) defined in the obvious way.)

\[
\begin{align*}
p & = \bigcup_{c=1}^{\infty} \text{DTIME}(n^c) \\
p_2 & = \bigcup_{c=1}^{\infty} \text{DTIME}(n^{\log^c n}) \\
p_{\text{space}} & = \bigcup_{c=1}^{\infty} \text{DSPACE}(n^c) \\
p_{2\text{space}} & = \bigcup_{c=1}^{\infty} \text{DSPACE}(n^{\log^c n}) \\
\text{rec} & = \{ f \mid f \text{ is computable} \}
\end{align*}
\]

The above \( \Delta \) classes induce measure structure on the \( R(\Delta) \) classes, where \( R(p) = \text{E} \), \( R(p_2) = \text{E}_2 \), \( R(p_{\text{space}}) = \text{ESPACE} \), \( R(p_{2\text{space}}) = \text{E}_2\text{SPACE} \), and \( R(\text{rec}) = \text{REC} \).

The following definitions formalize measure inside of the \( R(\Delta) \) classes.

**Definition** (Lutz [63]). Let \( X \subseteq \{0,1\}^\infty \) and let \( X^c \) denote the complement of \( X \).

1. A \( \Delta \)-null cover of \( X \) is a null cover of \( X \) that is \( \Delta \)-computable.
2. \( X \) has \( \Delta \)-measure 0, and we write \( \mu_\Delta(X) = 0 \), if there exists a \( \Delta \)-null cover of \( X \).
3. \( X \) has \( \Delta \)-measure 1, and we write \( \mu_\Delta(X) = 1 \), if \( \mu_\Delta(X^c) = 0 \).
4. \( X \) has measure 0 in \( R(\Delta) \), and we write \( \mu(X \mid R(\Delta)) = 0 \), if \( \mu_\Delta(X \cap R(\Delta)) = 0 \).
5. \( X \) has measure 1 in \( R(\Delta) \), and we write \( \mu(X \mid R(\Delta)) = 1 \), if \( \mu(X^c \mid R(\Delta)) = 0 \).

In this case, we say that \( X \) contains *almost every* language in \( R(\Delta) \).
In [63, 66], Lutz shows that the above definitions endow the $R(\Delta)$ classes with internal measure-theoretic structure. More specifically, Lutz shows that the collection $I_\Delta$ of all $\Delta$-measure 0 sets and the collection $I_{R(\Delta)}$ of all sets of measure 0 in $R(\Delta)$ are "$\Delta$-ideals," i.e., they are closed under subsets, finite unions, and "$\Delta$-unions." Moreover, Lutz shows that no cylinder $C_w$ has $\Delta$-measure 0 or measure 0 in $R(\Delta)$. Thus the $\Delta$-measure 0 sets and the sets of measure 0 in $R(\Delta)$ behave set theoretically as "small" sets.

The above definitions are sufficient to characterize the "small" and "large" sets within the $R(\Delta)$ complexity classes. However, the direct application of the above definitions is often too cumbersome for our needs. To remedy this situation, we mention two sufficient conditions for a set to have $\Delta$-measure 0. The first condition is that a $\Delta$-union of $\Delta$-measure 0 sets has $\Delta$-measure 0.

**Definition** (Lutz [63]). Let $Z, Z_0, Z_1, Z_2, \ldots \subseteq \{0, 1\}^\omega$. Then $Z$ is a $\Delta$-union of the $\Delta$-measure 0 sets $Z_0, Z_1, Z_2, \ldots$ if $Z = \bigcup_{j=0}^{\infty} Z_j$ and there exists a $\Delta$-computable 2-DS $d$ such that each $d_j$ is a $\Delta$-null cover of $Z_j$.

**Lemma 2.1** ([63]). Let $Z, Z_0, Z_1, Z_2, \ldots \subseteq \{0, 1\}^\omega$. If $Z$ is a $\Delta$-union of the $\Delta$-measure 0 sets $Z_0, Z_1, Z_2, \ldots$, then $Z$ has $\Delta$-measure 0.

The second condition is a resource-bounded version of the first Borel-Cantelli lemma. In order to state this condition, we require a resource-bounded notion of convergence for infinite series. (Notice that our series consist exclusively of nonnegative terms.) A *modulus* for a series $\sum_{n=0}^{\infty} a_n$ is a function $m : \mathbb{N} \to \mathbb{N}$ such that

$$\sum_{n=m(j)}^{\infty} a_n \leq 2^{-j}$$
for all $j \in \mathbb{N}$. A series is $\Delta$-convergent if it has a modulus $m \in \Delta$.

**Lemma 2.2** (Lutz[63]). If $d$ is a $\Delta$-computable 1-DS such that the series $\sum_{n=0}^{\infty} d_n(\lambda)$ is $\Delta$-convergent, then

\[ \mu_\Delta(\bigcap_{t=0}^{\infty} \bigcup_{n=t}^{\infty} S[d_n]) = \mu_\Delta(\{ A | A \in S[d_n] \text{ i.o.} \}) = 0. \]

### 2.4.1.2 Martingales

In Chapter 6 we find it convenient to work with a formulation of resource-bounded measure in terms of computable betting strategies called martingales. This version of resource-bounded measure is mentioned briefly in [63] and developed further in [66, 65].

**Definition** ([63]). A **martingale** is a function $d : \{0,1\}^* \rightarrow [0,\infty)$ with the property that, for all $w \in \{0,1\}^*$,

\[ d(w) = \frac{d(w0) + d(w1)}{2}. \quad (2.4) \]

A martingale succeeds on a sequence $x \in \{0,1\}^\infty$ if

\[ \lim_{n \rightarrow \infty} \sup d(x[0..n-1]) = \infty. \quad (2.5) \]

The set of sequences for which $d$ succeeds is denoted by

\[ S^{\infty}[d] = \{ x \in \{0,1\}^\infty \mid d \text{ succeeds on } x \}. \quad (2.6) \]

Intuitively, a martingale is a betting strategy which begins with a finite amount of money, $d(\lambda)$, and attempts to become infinitely wealthy. Condition (2.4) ensures that the betting is fair.
As in our previous version of resource-bounded measure, we approximate martingales via the dyadic rationals. Here, a computation of a martingale \( d \) is a function \( \hat{d} : \mathbb{N} \times \{0,1\}^* \to D \) that satisfies
\[
|\hat{d}_r(w) - d(w)| \leq 2^{-r}
\] (2.7)
for all \( r \in \mathbb{N} \) and \( w \in \{0,1\}^* \) such that \( r \geq |w| \). Moreover, a \( \Delta \)-computation of \( d \) is a computation \( \hat{d} \) of \( d \) such that \( \hat{d} \in \Delta \). These definitions allow us to characterize the "small" sets in terms of martingales.

**Theorem 2.3** (Lutz [63, 66]). A set \( X \subseteq \{0,1\}^\infty \) has \( \Delta \)-measure 0 if and only if there is a \( \Delta \)-computable martingale \( d \) such that \( X \subseteq S^\infty[d] \).

Theorem 2.3 provides a characterization of the \( \Delta \)-measure 0 sets in terms of martingales. Notice that this theorem is sufficient for a complete formalization of resource bounded measure in terms of martingales.

### 2.4.1.3 Pseudorandomness

We end this subsection with a brief discussion of resource-bounded randomness. In [63], it is shown that the definitions of resource-bounded measure admit a natural definition for what it means for an individual sequence \( x \in \{0,1\}^\infty \) to be random with respect to some resource bound.

**Definition** (Lutz [63]). A sequence \( x \in \{0,1\}^\infty \) is \( \Delta \)-random if there is no \( \Delta \)-computable martingale that succeeds on \( x \).

Here we write \( \text{RAND}(\Delta) \) for the set of all \( \Delta \)-random sequences. Notice that no language in \( R(\Delta) \) can be \( \Delta \)-random.

**Theorem 2.4** (Lutz [63]). \( \text{RAND}(\Delta) \cap R(\Delta) = \emptyset \).
However, notice that $\Delta$-random sequences are abundant in complexity classes that are larger than $R(\Delta)$.

**Theorem 2.5** (Lutz [63]).

1. $\mu(\text{RAND}(p)|E^2) = 1$.

2. $\mu(\text{RAND}(\text{pspace})|E^2\text{SPACE}) = 1$.

We use Theorems 2.4 and 2.5 in our investigation of the distribution of weakly complete languages in Chapter 6.

### 2.4.2 Baire Category

The theory of Baire category provides our final notion of the “size” of subsets of $\{0,1\}^\infty$. Briefly, Baire category classifies subsets of $\{0,1\}^\infty$ based on whether or not they can be represented as “countable unions of nowhere dense sets.” (See [78], for example.) A set $X \subseteq \{0,1\}^\infty$ is said to be *meager* if it can be represented as a countable union of nowhere dense sets. (Meager sets are also known as sets of *first category*.) A set $X$ is *not meager* (equivalently, $X$ is a *set of second category*) if it cannot be represented by a countable union of nowhere dense sets. A set $X$ is *co-meager* if $X^c$ is meager. (The co-meager sets are sometimes referred to as the *residual sets*.)

The above characterization of the meager sets in terms of “countable unions of nowhere dense sets” is the standard textbook definition, as found in [75, 78, 80, etc.]. Here we find it more convenient to use an equivalent characterization of the meager sets in terms of a two-person infinite game of perfect information, called the Banach-Mazur game. We use this characterization as *the* definition of the meager sets.
Roughly speaking, a Banach-Mazur game consists of a payoff set \( X \subseteq \{0,1\}^\infty \) and two players, Player I and Player II. The game is played in rounds with the two players taking turns extending a finite string. The game begins with Player I extending the empty string. The end result of the game is an infinite string \( x \in \{0,1\}^\infty \). Player I's goal is to force \( x \) to be in the payoff set \( X \). Player II's goal is to keep \( x \) out of \( X \). Player I wins if \( x \in X \) and Player II wins if \( x \notin X \).

More precisely, a play of a Banach-Mazur game consists of two competing strategies \( \sigma \) and \( \tau \). Each strategy is a function \( \sigma : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^* \) with the property that \( w_{\sigma_m(w)} \) for every \( w \in \{0,1\}^* \) and \( m \in \mathbb{N} \). Play begins with the empty string, \( w_0 = \lambda \). Player I extends \( w_0 \) by applying \( \sigma_0 \). Player II then extends \( w_1 = \sigma_0(w_0) \) by applying \( \tau_0 \). The game continues in this fashion forever. The play of \( (\sigma, \tau) \) generates an infinite sequence of strings \( w_0 \subseteq w_1 \subseteq w_2 \subseteq \ldots \) by the following recursion.

\[
\begin{align*}
w_0 &= \lambda, \\
w_{2m+1} &= \sigma_m(w_{2m}) \quad \text{for every } m \in \mathbb{N}, \\
w_{2m+2} &= \tau_m(w_{2m+1}) \quad \text{for every } m \in \mathbb{N}.
\end{align*}
\]

The result of the play \( (\sigma, \tau) \) is the sequence \( R(\sigma, \tau) \in \{0,1\}^\infty \) that satisfies

\[
R(\sigma, \tau)[n] = w_{n+1}[n]
\]

for every \( n \in \mathbb{N} \).

Let \( G[X] \) denote the Banach-Mazur game with payoff set \( X \subseteq \{0,1\}^\infty \). Then we say that a strategy \( \sigma \) is a winning strategy for Player I in \( G[X] \) if for all strategies \( \tau \), \( R(\sigma, \tau) \in X \). Similarly, we say that a strategy \( \tau \) is a winning strategy for Player II in \( G[X] \) if for all strategies \( \sigma \), \( R(\sigma, \tau) \notin X \). The above definitions allow us to characterize the meager sets in terms of Banach-Mazur games.
**Definition.** A set $X \subseteq \{0,1\}^\omega$ is **meager** if there exists a winning strategy for Player II in the Banach-Mazur game $G[X]$. A set $X \subseteq \{0,1\}^\omega$ is **co-meager** if $X^c$ is meager.

**Remark 2.6.** The proof that the above definitions are equivalent to the standard definitions of meager and co-meager is due to Banach and can be found in either [75] or [78].

The meager sets behave set-theoretically as small sets. For example, it is well-known [78] that the meager sets are closed under subsets and countable unions. Moreover, no cylinder is meager\(^2\) by the following simple argument. Let $w \in \{0,1\}^*$ and define a strategy $\sigma : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ by

$$\sigma_0(\lambda) = w$$
$$\sigma_n(y) = y1.$$

Then it is clear that $w \subseteq R(\sigma, \tau)$ for any possible strategy $\tau$ and hence that $R(\sigma, \tau) \in C_w$. It follows that there is no winning strategy for Player II in $G[C_w]$ and hence $C_w$ is not meager.

Since the meager sets behave set-theoretically as small sets, we say that the meager sets are **negligibly small in the sense of Baire category.** Similarly, we say that the co-meager sets are **large in the sense of Baire category.** We use these intuitive notions of “small” and “large” in our discussions of computational depth in Chapter 3.

Our three notions of “size” appear to adequately capture the intuitive notions of “small” and “large” subsets of $\{0,1\}^\omega$. However, it is easy to see that these notions

\(^2\)This result is the well-known Baire Category Theorem [78].
are not equivalent. In fact, a set may be large in one notion and yet small in the other notions. For example, consider the set RAND of all algorithmically random sequences. It is well-known [70] that RAND is meager and has measure 1. Moreover since RAND \cap REC is empty, it is clear that RAND has measure 0 in REC. Thus RAND is large in the sense of Lebesgue measure but small in both the senses of Baire category and resource-bounded measure.

The fact that certain sets may be “small” one sense and yet “large” in another does not present a problem here. For each of our applications, only one of the above notions of “size” provides a nontrivial measure. In Chapter 4-6, we consider sets of languages within the complexity classes (e.g., E, ESPACE, REC). Since all of our complexity classes are countable, each complexity class is meager and has measure 0. Therefore every subset of a complexity class is meager and has measure 0. So the measures provided by Baire category and Lebesgue measure are completely uninteresting in this case. For these reasons, we use resource-bounded measure as our measure of “size” in Chapters 4-6.

In Chapter 3 we use Baire category because neither Lebesgue measure nor resource-bounded measure provide an adequate measure for sets of computationally deep sequences. In Chapter 3 we show that no computationally deep sequence is algorithmically random or recursive. Therefore we are only interested in subsets of the set $D = \{0,1\}^\infty - (\text{RAND} \cup \text{REC})$. Since RAND has Lebesgue measure 1, the set $D$ has Lebesgue measure 0. Moreover since $D \cap \text{REC}$ is empty, $D$ has measure 0 in every complexity classes for which resource-bounded measure is defined. However, both RAND and REC are meager. It follows that the set $D$ is co-meager. Thus we

---

3RAND was first defined by Martin-Löf [70]. We review the definition of RAND in Section 2.5.
are able to distinguish between subsets of $D$ based on their "size" in the sense of Baire category.

## 2.5 Algorithmic Information and Randomness

Program-size complexity (commonly referred to as *Kolmogorov complexity*) was discovered independently by Solomonoff [87], Kolmogorov [44], and Chaitin [19]. Briefly, the Kolmogorov complexity of a string $x \in \{0,1\}^*$ is the length of the "shortest program that prints $x$." More precisely, if $x$ is a string and $M$ is a Turing machine with one input tape, one output tape, and $k$-worktapes, then the *Kolmogorov complexity* of $x$ with respect to $M$ is

$$C_M(x) = \min \{|\pi| \mid M(\pi) = x\}, \quad (2.8)$$

i.e., the length of the shortest program $\pi$ that produces $x$ when given as input to the machine $M$. (For all of our purposes, we assume that $\min \emptyset = \infty$.) It is well-known [57] that there are machines $U$ that are *universal* in the sense that for every machine $M$ there is a constant $c_M$ such that

$$C_U(x) \leq C_M(x) + c_M \quad (2.9)$$

for every string $x \in \{0,1\}^*$. Fix a universal machine $U$. Then the *Kolmogorov complexity* of $x$ is

$$C(x) = C_U(x). \quad (2.10)$$

Kolmogorov complexity and its variants have been used in a number of different applications in complexity theory. (See the recent book by Li and Vitanyi [57], for
example.) In Chapters 3 and 5 below, we use two variants of Kolmogorov complexity and two closely related notions. In Chapter 3, we employ self-delimiting Kolmogorov complexity (also known as *algorithmic information content* or *algorithmic entropy*) and the related notions of algorithmic probability and algorithmic randomness. In Chapter 5, we investigate the space-bounded Kolmogorov complexity of languages. In this section, we precisely define these notions and review their fundamental properties. We begin by reviewing our version of space-bounded Kolmogorov complexity.

### 2.5.1 Resource-bounded Kolmgorov Complexity

Resource-bounded versions of Kolmogorov complexity have been investigated extensively in the work of Kolmogorov [44], Hartmanis [31], Sipser [86], Levin [53], Longpré [58, 59], Balcázar and Book [6], Huynh [35], Ko [42], Allender [3], Allender and Rubinstein [2], Allender and Watanabe [4], Lutz [60, 61, 63], and many others. Here we use the space-bounded Kolmogorov complexity of languages, i.e., the space-bounded Kolmogorov complexity of the strings that represent the initial segments of languages. This variant of Kolmogorov complexity has been investigated by Huynh [35], Lutz [60, 61, 63] and others.

In our version of space-bounded Kolmogorov complexity, we assume that each machine $M$ uses two input tapes, one for the “program” and one for a natural number $n$. For a machine $M$ and “program” $\pi \in \{0,1\}^*$, we say that “$M(\pi,n) = w$ in $s$ space” if $M$, on input $(\pi,n)$, outputs the string $w \in \{0,1\}^*$ and halts without using more than $s$ cells of workspace. In Chapter 5 we are especially interested in situations where the output is of the form $\chi_{A_{=n}}$ or of the form $\chi_{A_{\leq n}}$, i.e., the $2^n$-bit characteristic string of $A_{=n}$ or the $(2^{n+1} - 1)$-bit characteristic string of $A_{\leq n}$, for some language $A$. 
Given a machine $M$, a space bound $s : \mathbb{N} \to \mathbb{N}$, a language $A \subseteq \{0,1\}^*$, and a natural number $n$, the $s(n)$-space-bounded Kolmogorov complexity of $A_{=n}$ relative to $M$ is

$$KS_M^{s(n)}(A_{=n}) = \min \{ |\pi| \mid M(\pi, n) = \chi_{A_{=n}} \text{ in } \leq s(n) \text{ space} \}.$$ 

Similarly, the $s(n)$-space-bounded Kolmogorov complexity of $A_{\leq n}$ relative to $M$ is

$$KS_M^{s(n)}(A_{\leq n}) = \min \{ |\pi| \mid M(\pi, n) = \chi_{A_{\leq n}} \text{ in } \leq s(n) \text{ space} \}.$$ 

Well-known simulation techniques show that there is a machine $U$ that is optimal in the sense that for each machine $M$ there is a constant $c$ such that for all $s, A$ and $n$, we have

$$KS_U^{c_s(n)+c}(A_{=n}) \leq KS_M^{s(n)}(A_{=n}) + c$$

and

$$KS_U^{c_{s(n)}+c}(A_{\leq n}) \leq KS_M^{s(n)}(A_{\leq n}) + c.$$ 

These facts are especially useful. Note that we fix an optimal machine $U$ and omit it from the notation.

### 2.5.2 Algorithmic Information Theory

In Chapter 3 we employ self-delimiting Kolmogorov complexity and the related notions of algorithmic probability and algorithmic randomness. These notions fall under the umbrella of algorithmic information theory. Here we define these notions and present their fundamental properties. The interested reader is directed to the books by Chaitin [21], Cover and Thomas [23], and Li and Vitányi [57] for a complete development of algorithmic information theory. We begin by defining self-delimiting
**Kolmogorov complexity**, a technical improvement over (2.8) that was formulated independently by Levin [48, 49], Schnorr [83], and Chaitin [20].

The self-delimiting Kolmogorov complexity of a string $x$ is the length of the "shortest program that prints $x$" in a restricted model of Turing machine computation. In this model of computation, the set of valid programs is restricted so that it forms an instantaneous code, i.e., if $\pi$ is a valid program in this model, then no proper prefix of $\pi$ is a valid program. Thus each program must be, in a sense, self-delimiting.

In our version of this model, each self-delimiting Turing machine has a single program tape, a single output tape, and $k$-worktapes. Each of the $k + 2$ tapes is
infinite. The program and output tapes are infinite to the right, and the \( k \)-worktapes are infinite in both directions. Only 0's, 1's, and blanks may appear on these tapes. See Figure 2.1 for an example of this type of restricted Turing machine.

Each of the \( k+2 \) tapes has a single dedicated scanning head. The scanning head for the program tape is read only and can only move to the right. The scanning head for the output tape is write only and can only move to the right. Moreover, the scanning head for the output tape is restricted to writing only 0's and 1's. The scanning heads for the \( k \)-worktapes may read or write and can move in either direction.

The initial state of a self-delimiting Turing machine \( M \) consists of its initial configuration with a program \( \pi \) on the program tape and the other tapes completely blank. In the initial state, the leftmost character of the program tape is blank and the program \( \pi \) lies immediately to the right of this character. In this state, the scanning heads for the program tape and output tape are positioned on the leftmost characters of their respective tapes.

The computation \( M(\pi) \) of a self-delimiting machine \( M \) on a program \( \pi \) begins with the machine \( M \) in its initial configuration and \( \pi \) on the program tape. The computation \( M(\pi) \) is a success, and we write \( M(\pi) \downarrow \), if \( M \) halts after finitely many steps with the program tape head scanning the last bit of \( \pi \). Otherwise, we say that the computation \( M(\pi) \) is a failure and write \( M(\pi) \uparrow \). If a computation \( M(\pi) \) is a success, then we write \( M(\pi) \in \{0,1\}^* \) for the contents of the output tape and \( time_M(\pi) \) for the number of steps that \( M \) took on \( \pi \). If the computation \( M(\pi) \) is a failure, then we assume that no output is produced and that \( time_M(\pi) = \infty \). If \( M(\pi) \downarrow \), then we say that \( \pi \) is a valid program for \( M \).

It is well-known [57] that there are self-delimiting Turing machines \( U \) that are
efficient and universal in the sense that for every machine $M$ there is a program prefix $\pi_M$ and a constant $c_M$ such that

(i) $U(\pi_M\pi) = M(\pi)$ and

(ii) $t(U(\pi_M\pi)) \leq c_M(1 + \text{time}_M(\pi) \log \text{time}_M(\pi))$

for all valid programs $\pi$ satisfying $M(\pi) \downarrow$. We fix one such efficient, universal self-delimiting Turing machine $U$ and define classical and time-bounded self-delimiting Kolmogorov complexity as follows.

**Definition.** Let $t : \mathbb{N} \rightarrow \mathbb{N}$ be a time-bound and let $M$ be a self-delimiting Turing machine.

1. The self-delimiting Kolmogorov complexity of $x \in \{0, 1\}^*$ relative to $M$ is

   $$K_M(x) = \min \{|\pi| \mid M(\pi) \downarrow \text{ and } M(\pi) = x\},$$

   the length of the shortest valid program for $M$ that produces $x$. The self-delimiting Kolmogorov complexity of $x \in \{0, 1\}^*$ is

   $$K(x) = K_U(x).$$

2. The $t$-time-bounded (self-delimiting) Kolmogorov complexity of $x$ relative to $M$ is

   $$K_M^t(x) = \min \{|\pi| \mid M(\pi) = x \text{ and } \text{time}_M(\pi) \leq t(|x|)\}.$$

   The $t$-time-bounded (self-delimiting) Kolmogorov complexity of $x$ is

   $$K^t(x) = K^t_U(x).$$

Self-delimiting Kolmogorov complexity is known here by several terms. In Chapter 1, we refer to self-delimiting Kolmogorov complexity as *algorithmic information*
In this section and in Chapter 3, we occasionally refer to self-delimiting Kolmogorov complexity as *algorithmic entropy*. These three terms are synonymous.

At this point we note a fundamental relationship between the algorithmic entropy of programs and the strings produced by these programs. The following result is well-known and obvious, but very useful.

**Lemma 2.7.** There is a constant $c_0 \in \mathbb{N}$ such that, for all $x \in \{0,1\}^*$ and all $\pi$ satisfying $U(\pi) = x$,

$$K(x) \leq K(\pi) + c_0.$$

Closely related to the notion of algorithmic entropy is the notion of algorithmic probability. This notion requires some additional notation. For a given self-delimiting Turing machine $M$ and string $x \in \{0,1\}^*$, we write

$$\text{PROG}_M = \{\pi \mid M(\pi) \downarrow\}$$

and

$$\text{PROG}_M(x) = \{\pi \mid M(\pi) \downarrow \text{ and } M(\pi) = x\}$$

for the set of valid programs for $M$ and the set of valid programs for $M$ that produce $x$, respectively. Let $t : \mathbb{N} \rightarrow \mathbb{N}$ be a time-bound and let $M$ be a self-delimiting Turing machine. Then, we write

$$\text{PROG}_M^t(x) = \{\pi \mid M(\pi) = x \text{ and } \text{time}_M(\pi) \leq t(|x|)\}$$

for the set of all $t$-time-bounded programs for $M$ that produce $x$. We write PROG for PROG$_U$, PROG$_M(x)$ for PROG$_U(x)$, and PROG$_M^t(x)$ for PROG$_U^t(x)$.

Since the scanning head for the program tape of a self-delimiting Turing machine may only move right, the sets PROG$_M$, PROG$_M(x)$, and PROG$_M^t(x)$ are all
instantaneous codes, i.e., in each of these sets no string is the prefix of any other. The above fact, in combination with the well-known Kraft inequality, implies that
\[ \sum_{\pi \in PROG_M} 2^{-|\pi|} \leq 1. \]
These observations allow us to precisely define the probability that "a fixed machine \( M \) produces the string \( x \) when given a random program," the algorithmic probability of \( x \).

For an instantaneous code \( I \subseteq \{0,1\}^* \), we define the probability of \( I \) to be
\[ Pr(I) = \sum_{x \in I} 2^{-|x|}. \]
The probability \( Pr(I) \) corresponds to the probability that a sequence \( x \in \{0,1\}^\infty \) is in the set
\[ \bigcup_{\omega \in I} C_\omega \]
in the usual random experiment in which the bits of \( x \) are chosen by the independent toss of a fair coin. This definition leads to a natural notion of algorithmic probability.

**Definition.** Let \( x \in \{0,1\}^* \), let \( \tau : \mathbb{N} \to \mathbb{N} \) be a time bound, and let \( M \) be a self-delimiting Turing machine. The algorithmic probability of \( x \) relative to \( M \) is
\[ \alpha_M(x) = Pr(PROG_M(x)). \]
The algorithmic probability of \( x \) is
\[ \alpha(x) = \alpha_\nu(x). \]

The \( t \)-time-bounded algorithmic probability of \( x \) relative to \( M \) is
\[ \alpha_M^t(x) = Pr(PROG_M^t(x)). \]
The \textit{t-time-bounded algorithmic probability} of \( x \) is
\[
m'(x) = m_U'(x).
\]

Intuitively, the quantity \( m(x) \) corresponds to the probability that the universal machine produces the string \( x \) when a random string is placed on its program tape. As the following two theorems illustrate, the notions of algorithmic probability and algorithmic entropy are closely related.

\textbf{Theorem 2.8} (Levin [48, 49], Chaitin [20]). There is a constant \( \bar{c} \in \mathbb{N} \) such that, for all \( X \in \{0,1\}^* \),
\[
- \log m(X) \leq K(X) < - \log m(x) + \bar{c}.
\]

The time-bounded versions of algorithmic entropy and algorithmic probability are also closely related. A straightforward modification of the proof of Theorem 2.8 yields the following time-bounded version. (The following theorem is also seen as an immediate corollary of Lemma 3 of [56].)

\textbf{Theorem 2.9.} Let \( t : \mathbb{N} \to \mathbb{N} \) be recursive.

1. For all \( x \in \{0,1\}^* \),
\[
- \log m'(x) \leq K'(x).
\]

2. There exist a recursive function \( t_1 : \mathbb{N} \to \mathbb{N} \) and a constant \( c_1 \in \mathbb{N} \) such that, for all \( x \in \{0,1\}^* \),
\[
K'(x) < - \log m'(x) + c_1.
\]

In our development of computation depth in Chapter 3, the above theorems are especially useful. We also use the following lemma and corollary, both due to Bennett.
For the lemma, say that a string \( \pi \in \{0,1\}^* \) computes a finite instantaneous code \( I \) if 
\[
U(\pi) = [x_0, \ldots, x_{n-1}]
\]
is a binary string that encodes an enumeration of the elements \( x_0, \ldots, x_{n-1} \) of \( I \) in some standard fashion.

**Lemma 2.10** (Bennett [11]). There is a constant \( c' \in \mathbb{N} \) such that, for all \( \pi \in \{0,1\}^* \), if \( \pi \) computes a finite instantaneous code \( I \), then for all \( x \in I \),
\[
K(x) \leq |x| + \log \Pr(I) + |\pi| + c'.
\]
(Note that \(-|x| \leq \log \Pr(I) \leq 0\), so that the bound becomes tighter as \( \Pr(I) \) becomes smaller.)

**Proof.** Let \( M \) be a Turing machine that, with program \( \pi \hat{\pi} \), where \( \pi \) computes a finite instantaneous code and \( \hat{\pi} \in \{0,1\}^* \), performs as indicated in Figure 2.2. (If the program for \( M \) is not of this form, then the computation is a failure.) Since \( U \) is a universal Turing machine, there is a program prefix \( \pi_M \in \{0,1\}^* \) such that, for all \( \pi \in \{0,1\}^* \), \( U(\pi_M \pi) = M(\pi) \). Let
\[
c' = |\pi_M| + 1.
\]

To see that \( c' \) has the desired property, let \( \pi \in \{0,1\}^* \) compute a finite instantaneous code \( I \). If \( I = \emptyset \), then the lemma is affirmed vacuously, so assume that \( I \neq \emptyset \). Let \( x_0, \ldots, x_{n-1} \) and \( k_0, \ldots, k_{n-1} \) be as in Figure 2.2. Define real numbers \( r_0 < \cdots < r_n \) by the recursion
\[
r_0 = 0, \quad r_{i+1} = r_i + 2^{-k_i},
\]
and note that
\[
r_n = \sum_{i=0}^{n-1} 2^{-k_i} = \sum_{x \in I} 2^{-|\log \Pr(I)| - |x|} \leq \Pr(I)^{-1} \sum_{x \in I} 2^{-|x|} = 1.
\]
begin
    simulate $U(\pi)$ to obtain $I$ (on a worktape) in the form
    $I = \{x_0, \ldots, x_{n-1}\}$,
    where $x_0, \ldots, x_{n-1}$ are in standard order;
    $\pi' := \lambda$;
    for $0 \leq i < n$ do
        begin
            if $i = 0$ then $w := 0^{k_i}$ else $w := next(w, k_i)$,
            where $k_i = |x_i| - \lceil \log \Pr(f) \rceil$ and $next(w, k_i)$ is the
            immediate lexicographic successor of the string $w1^{k_i-|w|}$;
            while $\pi' \sqsubseteq w$ do
                if $\pi' = w$ then output $x_i$ and halt
                else $\pi' := \pi'b$, where $b$ is the
                next bit on the program tape
        end
    end $M(\pi \widehat{=} \pi')$.

Figure 2.2: The Turing Machine $M$ used in the proof of Lemma 2.10.
Define strings \( \hat{x}, \ldots, \hat{x}_{n-1} \in \{0,1\}^* \) by
\[
\hat{x}_0 = 0^{k_0}, \quad \hat{x}_{i+1} = \text{next}(\hat{x}_i, k_{i+1}),
\]
where the function \( \text{next} \) is defined as in Figure 2.2. A routine induction on \( i \) shows that each \( \hat{x}_i \) is the standard \( k_i \)-bit binary representation of the natural number \( r_i \cdot 2^{k_i} \).
(The key point in the induction step is that, for \( 0 \leq i < n - 1 \), we have \( r_i + 2^{-k_i} = r_{i+1} \leq r_{n-1} < r_n \leq 1 \), so \( r_i \cdot 2^{k_i} < 2^{k_i} - 1 \). By the induction hypothesis, this means that \( \hat{x}_i \) does not consist entirely of 1's, so \( \hat{x}_{i+1} = \text{next}(\hat{x}_i, k_{i+1}) \) contains only \( k_{i+1} \) bits.) Moreover, it is easily checked that, for all \( 0 \leq i < n \), \( \hat{x}_i \) is the value assigned to \( w \) by \( M \) during iteration \( i \) of the for-loop, and that
\[
U(\pi \_M \hat{x}_i) = M(\pi \hat{x}_i) = x_i,
\]
whence
\[
K(x_i) \leq |\pi \_M \hat{x}_i| = k_i + |\pi| + c' - 1
\leq |x_i| + \log \Pr(I) + |\pi| + c'.
\]

Corollary 2.11. For every recursive function \( t : \mathbb{N} \rightarrow \mathbb{N} \) there exists a constant \( c^* \in \mathbb{N} \) such that, for all \( y \in \{0,1\}^* \) and all \( \pi \in \text{PROG}_t(y) \),
\[
K(\pi) \leq |\pi| + \log m(y) + K(y) + c^*.
\]
(Note that \( -|\pi| \leq \log m(y) \leq 0 \), so the bound becomes tighter as the time-bounded algorithmic probability of \( y \) becomes smaller.)
Proof. Let \( t : \mathbb{N} \to \mathbb{N} \) be recursive. Let \( M \) be a Turing machine that, with program \( \pi \in \{0,1\}^* \), does the following. First \( M \) simulates \( U(\pi) \). If this computation does not succeed, then \( M(\pi) \uparrow \). Otherwise, if \( U(\pi) = y \), then \( M \) simulates \( U(\pi') \) for \( t(|y|) \) steps for every string \( \pi' \in \{0,1\}^{\leq t(|y|)} \), and uses the result of this simulation to output an (encoded) enumeration \([\pi_0, \ldots, \pi_{n-1}]\) of the finite instantaneous code \( \text{PROG'}(y) \).

Since \( U \) is a universal Turing machine, there is a program prefix \( \pi_M \in \{0,1\}^* \) such that, for all \( \pi \in \{0,1\}^* \), \( U(\pi_M \pi) = M(\pi) \). Let

\[
c^* = |\pi_M| + c',
\]

where \( c' \) is the constant given by Lemma 2.10. For \( y \in \{0,1\}^* \), let \( \pi_y \) be a shortest element of \( \text{PROG}(y) \). Then, for all \( y \), the string \( \pi_M \pi_y \) computes the finite instantaneous code \( \text{PROG'}(y) \). It follows by Lemma 2.10 that, for all \( y \in \{0,1\}^* \) and \( \pi \in \text{PROG'}(y) \),

\[
K(\pi) \leq |\pi| + \log \Pr(\text{PROG'}(y)) + |\pi_M \pi_y| + c' = |\pi| + \log m'(y) + K(y) + c^*.
\]

\[\square\]

In Chapter 3 we examine sets of sequences that are related by the Kolmogorov complexity of their initial segments. Let \( g : \mathbb{N} \to [0, \infty) \) be a function and let \( t : \mathbb{N} \to \mathbb{N} \) be a recursive time bound. We define the classes

\[
K_{i.o.}[< g(n)] = \{ x \in \{0,1\}^\infty \mid K(x[0..n-1]) < g(n) \text{ i.o.} \}
\]

and

\[
K_{i.o.}'[< g(n)] = \{ x \in \{0,1\}^\infty \mid K'(x[0..n-1]) < g(n) \text{ i.o.} \}.
\]
In the above definitions, \( g(n) \) is used as a “threshold value” for the Kolmogorov complexity of the \( n \)-bit prefix of a sequence \( x \in \{0,1\}^\infty \). These classes contain every sequence whose Kolmogorov complexity falls below the threshold value for infinitely many prefixes.

The following theorem says that almost every recursive sequence has very high time-bounded Kolmogorov complexity almost everywhere. We use this result in the proof of the main result in Chapter 3.

**Theorem 2.12** (Lutz [63]). For every recursive bound \( f : \mathbb{N} \to \mathbb{N} \) and every real number \( 0 < \alpha < 1 \),

\[
\mu(\mathbf{K}_t^f < \alpha n | \text{REC}) = 0.
\]

(Theorem 2.12 follows immediately from Corollary 4.9 of [63]. Corollary 4.9 of [63] is stronger in several respects.)

### 2.5.3 Algorithmic Randomness

In Chapter 3 we examine infinite binary sequences that are *intrinsically random* to any algorithmic process. We conclude this section with a brief discussion of this notion of algorithmic randomness. Algorithmic randomness was originally defined by Martin-Löf [70] using constructive versions of ideas from measure theory. Under Martin-Löf’s original definition, a sequence is *random* if it passes all computable tests of randomness. Subsequently, Levin [48, 49], Schnorr [83], and Chaitin [20] showed that algorithmic randomness could be characterized in terms of self-delimiting Kolmogorov complexity. (Indeed, this was an important motivation for developing the self-delimiting formulation.) For the purposes of this dissertation, it is convenient
to use this characterization as the definition.

**Definition.** A sequence \( x \in \{0,1\}^\infty \) is *algorithmically random*, and we write \( x \in \text{RAND} \), if there is a constant \( k \in \mathbb{N} \) such that \( K(x[0..n-1]) > n - k \) a.e. That is,

\[
\text{RAND} = \bigcup_{k=0}^\infty K_{\text{lo.}}[< n - k]^c.
\]

Thus a sequence is *random* if almost every initial segment of it has high Kolmogorov complexity. The following theorem summarizes the elementary properties of \( \text{RAND} \) that are used in Chapter 3.

**Theorem 2.13** (Martin-Löf [70]). \( \text{RAND} \) is a measure 1 subset of \( \{0,1\}^\infty \) that is closed under finite variations and does not contain the characteristic sequence of any recursively enumerable set.
CHAPTER 3. COMPUTATIONAL DEPTH

Recently, Bennett [10, 11] extended algorithmic information theory to include notions of computational depth for binary strings and infinite binary sequences. In this chapter we review these notions and present two new results. The first result says that every weakly useful sequence is strongly deep. The second result says that, in the sense of Baire Category, almost every sequence has depth-like properties. The results in this chapter are joint work with James Lathrop and Jack Lutz [37].

For binary strings, Bennett’s notion of computational depth (also known as “logical depth” in [10, 11]) roughly corresponds to the amount of time required to produce a string from its shortest description. This notion is closely related to Adelman’s notion of “potential” [1] and Koppel’s notion of “sophistication” [47]. For infinite binary sequences, Bennett defines notions of strong and weak computational depth. Roughly speaking, a sequence $x \in \{0, 1\}^\infty$ is strongly deep if all of its initial segments are deep. A sequence $x \in \{0, 1\}^\infty$ is weakly deep if it is not reducible in recursively bounded time to any random sequence. (See [10, 11, 37] for the intuition and motivation behind these notions.)

Some of the intuition behind Bennett’s notions of strong and weak computational depth lies in the observation that truly intricate sequences should be neither random nor simple. To justify this intuition, Bennett shows that no recursive or algo-
rithmically random sequence is strongly or weakly deep. Similarly, Bennett justifies
the intuition that strong depth captures a measure of computational usefulness by
showing that \( K \), the diagonal halting language, is strongly deep.

In this chapter we rigorously confirm Bennett's intuition about the notions of
strong and weak computational depth. In section 3.1 we examine Bennett's notion
of strong computational depth. There we show that no recursive or algorithmically
random sequence is strongly deep. Furthermore, we show that every weakly useful se-
quence is strongly deep. This result extends Bennett's observation that \( K \) is strongly
depth.

In section 3.2, we examine Bennett's notion of weak computational depth. There
we show that, in the sense of Baire Category, almost every sequence, is weakly deep.

3.1 Strong Computational Depth

In this section we examine Bennett's notion of strong computational depth for
infinite binary sequences. As shown in Theorem 3.4, this notion admits several equivalent characterizations. We begin this section by presenting a formulation of strong
depth in terms of parameterized depth classes.

**Definition.** For \( t, g : \mathbb{N} \to \mathbb{N} \) and \( n \in \mathbb{N} \), we define the sets

\[
D^t_g(n) = \{ x \in \{0, 1\}^\infty \mid (\forall \pi \in \text{PROG}'(x[0..n-1])) K(\pi) \leq |\pi| - g(n) \}
\]

and

\[
D^t_g = \bigcup_{n=0}^{\infty} \bigcap_{m=0}^{n} D^t_g(n) = \{ x \in \{0, 1\}^\infty \mid x \in D^t_g(n) \text{ a.e.} \}.
\]
A sequence $x \in \{0,1\}^\infty$ is *strongly deep*, and we write $x \in \text{strDEEP}$, if for every recursive time bound $t : \mathbb{N} \to \mathbb{N}$ and every constant $c \in \mathbb{N}$, $x \in D_c^t$.

We begin our examination of strong computational depth by reaffirming Bennett's observation that no strongly deep sequence can be algorithmically random. For this purpose, we require the following technical lemma concerning algorithmically random sequences.

**Lemma 3.1.** If $x \in \text{RAND}$, then there exist a sequence $k_0, k_1, \ldots$ of natural numbers and a sequence $\pi_0, \pi_1, \ldots$ of programs satisfying the following three conditions for all $i \in \mathbb{N}$.

1. For all $n \geq k_i$, $K(x[0..k_i - 1]) - k_i \leq K(x[0..n - 1]) - n$.
2. $U(\pi_i) = x[0..k_i - 1]$ and $|\pi_i| = K(x[0..k_i - 1])$.
3. $k_{i+1} > k_i + \text{time}_U(\pi_i)$.

**Proof.** Let $x \in \text{RAND}$. Define $f : \mathbb{N} \to \mathbb{Z}$ by $f(n) = K(x[0..n - 1]) - n$. For each $i \in \mathbb{N}$, fix the least argument $n_i \geq i$ such that $f(n_i) \leq f(n)$ for all $n \geq i$. (Since $x \in \text{RAND}$, $f$ is bounded below, so $n_i$ exists.) Define the sequences $k_0, k_1, \ldots$ and $\pi_0, \pi_1, \ldots$ recursively as follows. Let $k_0 = n_0$ and let $\pi_0$ be a minimal program for $x[0..k_0 - 1]$. Given $k_i$ and $\pi_i$, let $k_{i+1} = n_{k_i+\text{time}_U(\pi_i)+1}$ and let $\pi_{i+1}$ be a minimal program for $x[0..k_{i+1} - 1]$. It is easily verified that the sequences $k_0, k_1, \ldots$ and $\pi_0, \pi_1, \ldots$ satisfy conditions (1), (2), and (3). \qed

**Theorem 3.2** (Bennett [11]). $\text{RAND} \cap \text{strDEEP} = \emptyset$. In fact, there exist a recursive function $t(n) = O(n \log n)$ and a constant $c \in \mathbb{N}$ such that $\text{RAND} \cap D_c^t = \emptyset$. 
**Proof.** Let $M$ be a Turing machine that, with program $\pi y$, does the following. The machine $M$ simulates $U(\pi)$, recording $\text{time}_U(\pi)$ while doing so. If the simulated computation succeeds, $M$ then reads and outputs the first $\text{time}_U(\pi)$ bits of $y$ (appended to the string $U(\pi)$ already produced as output) and halts. Note that if $|y| = \text{time}_U(\pi)$, then the computation of $M(\pi y)$ succeeds, with $M(\pi y) = U(\pi)y$. Otherwise, the computation of $M(\pi y)$ is a failure.

On successful computations, the Turing machine $M$ takes $O(|y|)$ steps to produce $U(\pi)y$. Thus there exist a program prefix $\pi_M$ and a recursive, nondecreasing time bound $t(n) = O(n \log n)$ such that, for all successful computations $U(\pi)$ and all strings $y$ with $|y| = \text{time}_U(\pi)$, the following two conditions hold.

(i) $U(\pi_M\pi y) = U(\pi)y$.

(ii) $\text{time}_U(\pi_M\pi y) \leq t(|y|)$.

Let $c = |\pi_M| + c_0$, where $c_0$ is the constant from Lemma 2.7. We prove that $\text{RAND} \cap D^c = \emptyset$.

Let $x \in \text{RAND}$. Fix sequences $\pi_0, \pi_1, \ldots$ and $\pi_0, \pi_1, \ldots$ as in Lemma 3.1. For each $i \in \mathbb{N}$, let $n_i = k_i + \text{time}_U(\pi_i)$. Note that the sequence $n_0, n_1, \ldots$ is strictly increasing. We prove that $x \notin D^c$ by showing that, for all $i \in \mathbb{N}$, $x \notin D^c(n_i)$.

Conditions (i) and (ii) above imply that the following conditions hold for all $i \in \mathbb{N}$.

(iii) $U(\pi_M\pi_i x[k_i..n_i - 1]) = x[0..n_i - 1]$.

(iv) $\text{time}_U(\pi_M\pi_i x[k_i..n_i - 1]) \leq t(n_i - k_i) \leq t(n_i)$.

Then, for all $i \in \mathbb{N}$

$$\pi_M\pi_i x[k_i..n_i - 1] \in \text{PROG}'(x[0..n_i - 1])$$
and Lemma 3.1 tells us that

\[ K(x[0..k_i - 1]) \leq K(x[0..n_i - 1]) - n_i + k_i = K(x[0..n_i - 1]) - \text{time}_U(\pi_i), \]

whence

\[ K(\pi_x x[k_i..n_i - 1]) \geq K(x[0..n_i - 1]) - c_0 \]
\[ \geq K(x[0..k_i - 1]) + \text{time}_U(\pi_i) - c_0 \]
\[ = |\pi_x| + n_i - k_i - c_0 \]
\[ = |\pi_x x[k_i..n_i - 1]| - c_0 \]
\[ = |\tau x[k_i..n_i - 1]| - c. \]

Thus \( x \not\in D^i_c(n_i) \) for all \( i \in \mathbb{N} \), so \( x \not\in D^i_c \).

As mentioned above, the notion of strong computational depth has several equivalent characterizations. Here we show that our definition is equivalent to Bennett's original definition and two other definitions based on algorithmic entropy and algorithmic probability. We first recall Bennett's original definition of computational depth for binary strings.

**Definition** ([11]). Let \( w \in \{0,1\}^* \) and \( c \in \mathbb{N} \). Then the *computational depth of \( w \) at significance level \( c \) is*

\[ \text{depth}_c(w) = \min\{t \in \mathbb{N} \mid (\exists \pi \in \text{PROG}^t(w)) |\pi| < K(x) + c\}. \]

Using the above definition, Bennett defines a sequence \( x \) to be strongly deep if for every recursive time bound \( t : \mathbb{N} \to \mathbb{N} \) and every constant \( c \in \mathbb{N} \), \( \text{depth}_c(x[0..n - \)
This definition is easily seen to be equivalent to one in terms of the parameterized depth classes $D_i^t$. We now present two related parameterized depth classes defined in terms of algorithmic entropy and algorithmic probability.

**Definition.** For $t, g : \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, we define the sets

\[
D_g^t(n) = \{ x \in \{0,1\}^\infty \mid K(x[0..n - 1]) \leq K^t(x[0..n - 1]) - g(n) \},
\]

\[
\tilde{D}_g^t(n) = \{ x \in \{0,1\}^\infty \mid m(x[0..n - 1]) \geq 2^s(n)m^t(x[0..n - 1]) \},
\]

\[
\tilde{D}_g^t(n) = \bigcup_{m=0}^{\infty} \bigcap_{n=m}^{\infty} \tilde{D}_g^t(n),
\]

\[
\tilde{D}_g^t(n) = \bigcup_{m=0}^{\infty} \bigcap_{n=m}^{\infty} \tilde{D}_g^t(n).
\]

The following technical lemma illustrates the relationships among the various parameterized depth classes. (Note that lemma 3.3 was proven in a different form in [11].)

**Lemma 3.3** (Bennett [11]). If $t : \mathbb{N} \to \mathbb{N}$ is recursive, then there exist constants $c_0, c_1, c_2 \in \mathbb{N}$ and a recursive function $t_1 : \mathbb{N} \to \mathbb{N}$ such that the following six conditions hold for all $g : \mathbb{N} \to \mathbb{N}$ and all $n \in \mathbb{N}$.

1. $D_g^{t+1}(n) \subseteq \tilde{D}_g^t(n)$
2. $\tilde{D}_g^{t+1}(n) \subseteq \tilde{D}_g^t(n)$
3. $\tilde{D}_g^{t+1}(n) \subseteq D_g^t(n)$
4. $D_g^{t+c_0} \subseteq \tilde{D}_g^t$
5. $\tilde{D}_g^{t+c_1} \subseteq \tilde{D}_g^t$
6. $\tilde{D}_g^{t+c_2} \subseteq D_g^t$

**Proof.** It suffices to prove 1, 2, and 3, since 4, 5, and 6 then follow immediately.

1. Let $c_0$ be as in Lemma 2.7 and assume that $x \in D_g^{t+c_0}(n)$. Let $\pi$ be a shortest element of $\text{PROG}^t(x[0..n - 1])$. Since $x \in D_g^{t+c_0}(n)$, we have $K(\pi) \leq |\pi| - g(n) - c_0$. 
It follows that

\[
K(x[0..n-1]) \leq K(\pi) + c_0 \\
\leq |\pi| - g(n) \\
= K^t(x[0..n - 1]) - g(n),
\]

whence \( x \in \bar{D}_g^t(n) \).

2. Choose \( c_1 \) and \( t_1 \) for \( t \) as in Theorem 2.9 and assume that \( x \in \bar{D}_{g+c_1}(n) \). Then

\[
K(x[0..n-1]) \leq K^{t_1}(x[0..n-1]) - g(n) - c_1.
\]

It follows by Theorems 2.8 and 2.9 that

\[
m(x[0..n-1]) \geq 2^{-K(x[0..n-1])} \\
\geq 2^{g(n) + c_1 - K^{t_1}(x[0..n-1])} \\
> 2^{g(n)} m^t(x[0..n-1]),
\]

whence \( x \in \bar{D}_g^t(n) \).

3. Let \( \bar{c} \) be as in Theorem 2.8, choose \( c^* \) for \( t \) as in Corollary 2.11, let \( c_2 = \bar{c} + c^* \), and assume that \( x \in \bar{D}_{g+c_2}(n) \). Then

\[
K(x[0..n-1]) \leq -\log m(x[0..n-1]) + \bar{c} \\
\leq -\log m^t(x[0..n-1]) - g(n) - c_2 + \bar{c} \\
= -\log m^t(x[0..n-1]) - g(n) - c^*.
\]

Thus, for all \( \pi \in \text{PROG}^t(x[0..n-1]) \),

\[
K(\pi) \leq |\pi| + K(x[0..n-1]) + \log m^t(x[0..n-1]) + c^* \\
\leq |\pi| - g(n),
\]

whence \( x \in D_g^t(n) \). \( \square \)
We use Lemma 3.3 to show that strong computational depth has several equivalent characterizations.

**Theorem 3.4** (Bennett [11]). For \( x \in \{0,1\}^\infty \), the following four conditions are equivalent.

1. \( x \) is strongly deep.
2. For every recursive time bound \( t : \mathbb{N} \to \mathbb{N} \) and every constant \( c \in \mathbb{N} \),
   \[ \text{depth}_c(x[0..n-1]) > i(n) \text{ a.e.} \]
3. For every recursive time bound \( t : \mathbb{N} \to \mathbb{N} \) and every constant \( c \in \mathbb{N} \), \( x \in D^t_c \).
4. For every recursive time bound \( t : \mathbb{N} \to \mathbb{N} \) and every constant \( c \in \mathbb{N} \), \( x \in D^t_c \).

**Proof.** The equivalence of (1) and (2) follows immediately from the definitions. The equivalence of (1), (3), and (4) follows immediately from Lemma 3.3. \( \square \)

Note that, as mentioned previously, (1) and (2) are easily seen to be equivalent. In [11], Bennett uses (2) as the definition of strong computational depth and implicitly proves the equivalence of (2), (3), and (4). Li and Vitányi [55, 57] essentially use (4) as the definition in their discussion of computational depth. Here we use these characterizations of strong computational depth interchangeably.

Our next lemma essentially shows that depth cannot be generated quickly by deterministic processes. This technical lemma is related to Bennett’s slow growth law [11]. For this lemma, we require some specific notation. First, for any function \( s : \mathbb{N} \to \mathbb{N} \), we define the function \( s^* : \mathbb{N} \to \mathbb{N} \) by

\[ s^*(n) = 2^{s([\log n])+1}. \]
Second, for any unbounded, nondecreasing function $f : \mathbb{N} \to \mathbb{N}$, we define the special-purpose "inverse" function $f^{-1} : \mathbb{N} \to \mathbb{N}$ by

$$f^{-1}(n) = \max\{m \mid f(m) < n\}.$$  

Also, for this lemma, say that a function $s : \mathbb{N} \to \mathbb{N}$ is \emph{time-constructible} if there exist a constant $c_s \in \mathbb{N}$ and a Turing machine that, given the standard binary representation $w$ of a natural number $n$, computes the standard binary representation of $s(n)$ in at most $c_s \cdot s(|w|)$ steps. Using standard techniques [33, 7], it is easy to show that, for every recursive function $r : \mathbb{N} \to \mathbb{N}$, there is a strictly increasing, time-constructible function $s : \mathbb{N} \to \mathbb{N}$ such that, for all $n \in \mathbb{N}$, $r(n) \leq s(n)$.

\textbf{Lemma 3.5.} Let $s : \mathbb{N} \to \mathbb{N}$ be strictly increasing and time-constructible, with the constant $c_s \in \mathbb{N}$ as witness. For each $s$-time-bounded oracle Turing machine $M$, there is a constant $c_M \in \mathbb{N}$ with the following property. Given nondecreasing functions $t, g : \mathbb{N} \to \mathbb{N}$, define the functions $\tau, \hat{t}, \hat{g} : \mathbb{N} \to \mathbb{N}$ by

$$\tau(n) = t(s^*(n + 1)) + 4s^*(n + 1) + 2(n + 1)c_s s(l) + 2ns^*(n + 1)s(l),$$

$$\hat{t} = c_M (1 + \tau(n)[\log \tau(n)]),$$

$$\hat{g} = g(s^*(n + 1)) + c_M,$$

where $l$ is the number of bits in the binary representation of $n$. For all $x, y \in \{0, 1\}^\omega$, if $y \leq_{\text{DTIME}(s)} x$ via $M$ and $y \in D^f_{\hat{g}}$, then $x \in D^f_{\hat{g}}$.

\textbf{Proof.} Let $s$ and $M$ be as in the statement of the lemma. Let $M'$ be a Turing machine that, with program $\pi \in \{0, 1\}^*$, operates as in Figure 3.1. Since $U$ is an efficient universal Turing machine, there exist a program prefix $\pi_{M'} \in \{0, 1\}^*$ and a
constant $c_{M'} \in \mathbb{N}$ such that, for all $\pi \in \{0,1\}^*$,

$$U(\pi_{M',\pi}) = M'({\pi})$$

and

$$time_U(\pi_{M',\pi}) \leq c_{M'}(1 + time_{M'}({\pi}) \log time_{M'}({\pi}))$$

Let $M''$ be a Turing machine that, with program $\pi^* \in \{0,1\}^*$, simulates $U(\pi^*)$ and outputs $\pi$ if and only if $U(\pi^*) = \pi_{M',\pi}$. Since $U$ is universal, there is a program prefix $\pi_{M''} \in \{0,1\}^*$ such that, for all $\pi^* \in \{0,1\}^*$, $U(\pi_{M''\pi^*}) = M''(\pi^*)$. Let

$$c_M = \max \{c_{M'}, |\pi_{M'}| + |\pi_{M''}|\}.$$

Fix $m_0 \in \mathbb{N}$ such that $(s^*)^{-1}(m) > 0$ for all $m \geq m_0$.

begin
\[ u := U(\pi); \]
\[ n := (s^*)^{-1}(|u|); \]
\[ \text{for } 0 \leq i < n \text{ do } \]
\[ \quad \text{append the bit } M"^{0^{|u|}}(i) \text{ to the output; } \]
\[ \quad \text{halt; } \]
end $M'(\pi)$.

Figure 3.1: The Turing machine $M'$ used in the proof of Lemma 3.5.

Now define $\tau$, $\hat{\tau}$, and $\hat{g}$ as in the statement of the lemma and assume that $x, y \in \{0,1\}^\omega$ satisfy $y \leq^D_T x$ via $M$ and $y \in D^L_{\hat{g}}$. Fix $n_0 \in \mathbb{N}$ such that $y \in D^L_{\hat{g}}(n)$ for all $n \geq n_0$ and let

$$m_1 = \max \{m_0, s^*(n_0) + 1\}.$$

The following two claims are verified at the end of this proof.
Claim 1. For all $m \geq m_0$ and $\pi \in \{0,1\}^*$, if $\pi \in \text{PROG}^t(x[0..m - 1])$, then $\pi \cdot \pi \in \text{PROG}^t(y[0..n - 1])$, where $n = (s^*)^{-1}(m)$.

Claim 2. For all $m \geq m_1$ and all $\pi \in \text{PROG}^t(x[0..m - 1])$,

$$K(\pi) \leq |\pi| - \tilde{g}(n) + c_M,$$

where $n = (s^*)^{-1}(m)$.

To finish proving the lemma, let $m \geq m_1$ and let $\pi \in \text{PROG}^t(x[0..m - 1])$. Then, by Claim 2 and the monotonicity of $g$,

$$K(\pi) \leq |\pi| - \tilde{g}((s^*)^{-1}(m)) + c_M$$

$$= |\pi| - g(s^*((s^*)^{-1}(m) + 1))$$

$$\leq |\pi| - g(m).$$

Thus $x \in D_g(m)$. Since this holds for all $m \geq m_1$, it follows that $x \in D'_g$, affirming the lemma. All that remains, then, is to prove the two claims.

To prove Claim 1, assume that $m \geq m_0$ and $\pi \in \text{PROG}^t(x[0..m - 1])$. Let $u = x[0..m - 1]$ and $n = (s^*)^{-1}(m)$. Since $m \geq m_0$, we must have $s^*(n) < m$. Since $M$ is $s$-time-bounded, this implies that $M^{s^{*\infty}}(i) = M^{s}(i) = y[i]$ for all $0 \leq i < n$. (All queries in these computations must be to bits $x[j]$ for $j < |u|$.) Thus

$$U(\pi \cdot \pi) = M'(\pi) = y[0..n - 1].$$

With program $\pi$, $M'$ requires at most $t(m)$ steps to compute $u$, at most $4m$ additional steps to compute $|u|$ in binary, at most $2(n + 1)c_s s(l)$ steps to compute $n$, and at most $2nms(s(l)$ steps to execute the for-loop. Since $s^*(n + 1) \geq m$, and $t$ is nondecreasing,
it follows that $\text{time}_{M'}(\pi) \leq \tau(n)$, so

$$\text{time}_U(\pi_{M'}\pi) \leq \tilde{f}(n).$$

Thus $\pi_{M'}\pi \in \text{PROG}^g(y[0..n-1])$. This proves Claim 1.

Finally, to prove Claim 2, let $m \geq m_0$, let $\pi \in \text{PROG}'(x[0..m-1])$, and let $n = (s^*)^{-1}(m)$. Since $m > s^*(n_0)$, it must be the case that $n = (s^*)^{-1}(m) \geq n_0$, whence $y \in D_\tilde{g}(n)$. Since $m \geq m_0$, Claim 1 tells us that $\pi_{M'}\pi \in \text{PROG}^g(y[0..n-1])$. Since $y \in D_\tilde{g}$, it follows that

$$K(\pi_{M'}\pi) \leq |\pi_{M'}\pi| - \tilde{g}(n) = |\pi| - \tilde{g}(n) + |\pi_{M'}|.$$

Now let $\pi^*$ be a shortest element of $\text{PROG}(\pi_{M'}\pi)$. Then $U(\pi^*) = \pi_{M'}\pi$, so

$$U(\pi_{M''}\pi^*) = M''(\pi^*) = \pi,$$

so

$$K(\pi) \leq |\pi_{M''}\pi^*|$$

$$= K(\pi_{M'}\pi) + |\pi_{M''}|$$

$$\leq |\pi| - \tilde{g}(n) + c_M.$$

This proves Claim 2 and completes the proof of Lemma 3.5.

Technical Lemma 3.5 allows us to show that no strongly deep sequence is truth-table reducible to a sequence that is not also strongly deep. This implies that strong depth is invariant under truth-table equivalence, a fact noted by Bennett [11].

**Theorem 3.6.** Let $x, y \in \{0,1\}^\infty$. If $y \leq_U x$ and $y$ is strongly deep, then $x$ is strongly deep.
Proof. Assume the hypothesis. To see that \( x \) is strongly deep, fix a recursive function \( t : \mathbb{N} \to \mathbb{N} \) and a constant \( c \in \mathbb{N} \). It suffices to prove that \( x \in D_c^* \).

Since \( y \leq_T x \), there exist a strictly increasing time-constructible function \( s : \mathbb{N} \to \mathbb{N} \) and an \( s \)-time-bounded oracle Turing machine \( M \) such that \( y \leq_{\text{DTIME}(s)} x \) via \( M \). Choose a constant \( c_M \) for \( M \) as in Lemma 3.5 and define \( g : \mathbb{N} \to \mathbb{N} \) by \( g(n) = c \) for all \( n \in \mathbb{N} \). Then, in the notation of Lemma 3.5, \( \hat{g} \) is recursive and \( \tilde{g} \) is constant. Since \( y \) is strongly deep, it follows that \( y \in D_{\tilde{g}}^* \). It follows by Lemma 3.5 that \( x \in D_c^* \).

Theorem 3.6 allows us to reaffirm Bennett's observation that no recursive sequence is strongly deep.

**Corollary 3.7** (Bennett [11]). \( \text{REC} \cap \text{strDEEP} = \emptyset \).

Proof. Let \( x \in \text{REC} \); it suffices to show that \( x \not\in \text{strDEEP} \). Fix \( z \in \text{RAND} \). Then, trivially, \( x \leq_T z \). By Theorem 3.2, \( z \not\in \text{strDEEP} \), so by Theorem 3.6, \( x \not\in \text{strDEEP} \).

Although no recursive sequence is strongly deep, the following technical theorem tells us that every recursive sequence is either somewhat deep or somewhat compressible. (Recall from Chapter 2 that

\[
K_{i.o.}[< g(n)] = \{ x \in \{0,1\}^\infty \mid K^t(x[0..n - 1]) < g(n) \text{ i.o.} \}
\]

**Theorem 3.8.** If \( t : \mathbb{N} \to \mathbb{N} \) is recursive and \( 0 < \alpha < \beta < 1 \), then

\[
\text{REC} \subseteq \hat{D}_\alpha^t \cup K_{i.o.}[< \beta n] \ni
\]
Proof. Assume the hypothesis and let

\[ x \in \text{REC} - \mathbb{K}_{\omega}[< \beta n]. \]

It suffices to prove that \( x \in \hat{D}^t_{\alpha n} \).

Since \( x \notin \mathbb{K}_{\omega}[< \beta n] \), we have

\[ K'(x[0..n-1]) \geq \beta n \text{ a.e.} \]

Since \( x \) is recursive, it follows that there is a constant \( c \in \mathbb{N} \) such that, for all sufficiently large \( n \),

\[ K(x[0..n-1]) < 2\log n + c \]
\[ < \beta n - \alpha n \]
\[ \leq K'(x[0..n-1]) - \alpha n, \]

whence \( x \in \hat{D}^t_{\alpha n} \).

Theorem 3.8 immediately implies that recursive sequences exist (and are abundant) in arbitrarily high levels of the parameterized depth hierarchy.

Corollary 3.9. For every recursive function \( t : \mathbb{N} \rightarrow \mathbb{N} \) and every \( 0 < \gamma < 1 \), the set \( D^t_{\gamma n} \) has measure 1 in \( \text{REC} \).

Proof. Let \( t : \mathbb{N} \rightarrow \mathbb{N} \) be recursive and let \( 0 < \gamma < \alpha < \beta < 1 \). Choose a recursive function \( t_1 : \mathbb{N} \rightarrow \mathbb{N} \) and constants \( c_1, c_2 \in \mathbb{N} \) for \( t \) as in Lemma 3.3, so that

\[ \hat{D}^t_{\gamma n+c_2+c_1}(n) \subseteq \hat{D}^t_{\gamma n+c_2}(n) \subseteq D^t_{\gamma n}(n) \]

for all \( n \in \mathbb{N} \). For all sufficiently large \( n \),

\[ \hat{D}^t_{\alpha n}(n) \subseteq \hat{D}^t_{\gamma n+c_2+c_1}(n), \]
so it follows that \( \tilde{D}^t_{\alpha n} \subseteq D^t_{\gamma n} \).

By Theorem 2.12, \( K_{1,\omega}^{1,\omega} \) has measure 0 in \( \text{REC} \). By Theorem 3.8, this implies that \( \tilde{D}^t_{\alpha n} \) has measure 1 in \( \text{REC} \). Since \( \tilde{D}^t_{\alpha n} \subseteq D^t_{\gamma n} \), it follows that \( D^t_{\gamma n} \) has measure 1 in \( \text{REC} \). \( \square \)

**Corollary 3.10.** For every recursive function \( t : \mathbb{N} \to \mathbb{N} \) and every constant \( c \in \mathbb{N} \), \( D^t_c \) has measure 1 in \( \text{REC} \).

The previous theorems in this section provide sufficient machinery to prove the main result of this section, namely, that every weakly useful sequence is strongly deep. We first precisely define this notion of weak usefulness.

**Definition.** A sequence \( x \in \{0,1\}^\omega \) is weakly useful if there is a recursive time bound \( s : \mathbb{N} \to \mathbb{N} \) such that \( \text{DTIME}^x(s) \) does not have measure 0 in \( \text{REC} \).

**Theorem 3.11.** Every weakly useful sequence is strongly deep.

**Proof.** Let \( x \in \{0,1\}^\omega \) be weakly useful. To see that \( x \) is strongly deep, let \( t : \mathbb{N} \to \mathbb{N} \) be a recursive time bound, and let \( c \in \mathbb{N} \). It suffices to prove that \( x \in D^t_c \).

Since \( x \) is weakly useful, there is a recursive time bound \( s : \mathbb{N} \to \mathbb{N} \) such that \( \text{DTIME}^x(s) \) does not have measure 0 in \( \text{REC} \). Since every recursive function is bounded above by a strictly increasing, time-constructible function, we can assume without loss of generality that \( s \) is strictly increasing and time-constructible.

Let \( \tilde{t}(n) = n \cdot (1 + \tau(n) \lfloor \log \tau(n) \rfloor) \), where \( \tau \) is defined from \( t \) and \( s \) as in Lemma 3.5, and let \( \gamma = \frac{1}{2} \). Since \( \tilde{t} \) is recursive, Corollary 3.9 tells us that \( D^t_{\gamma n} \) has measure 1 in \( \text{REC} \). Since \( \text{DTIME}^x(s) \) does not have measure 0 in \( \text{REC} \), it follows that \( D^t_{\gamma n} \cap \)
DTIME^e(s) \neq \emptyset. Fix a sequence y \in D^\tilde{t}_{\gamma n} \cap DTIME^e(s). Then there is an s-time-bounded oracle Turing machine M such that y \leq^{DTIME(s)}_T x. Fix a constant c_M for M as in Lemma 3.5. Define g(n) = c for all n \in \mathbb{N} and define the functions \tau, \tilde{t}, and \tilde{g} from t and g as in Lemma 3.5. Since \tilde{g} and c_M are constant, we have \tilde{t}(n) > \tilde{t}(n) \text{ a.e.} and \gamma n > \tilde{g}(n) \text{ a.e.}, so y \in D^\tilde{t}_{\gamma n} \subseteq D^\tilde{g}_{\gamma n}. It follows by Lemma 3.5 that x \in D^t_{\epsilon}.

Theorem 3.11 immediately implies Bennett's observation that that K is strongly deep.

**Notation.** Let \chi_H and \chi_K be the characteristic sequences of the halting problem and the diagonal halting problem, respectively. That is, the sequences \chi_H, \chi_K \in \{0,1\}^\omega are defined by

\[
\chi_H[i,n] = 1 \iff M_i(n) \text{ halts},
\]

\[
\chi_K[n] = 1 \iff M_n(n) \text{ halts},
\]

where \(M_0, M_1, \ldots\) is a standard enumeration of all deterministic Turing machines and \((, )\) is a standard pairing function, e.g., \((i, n) = \left(i + n + 1\right) + n\).

**Corollary 3.12** (Bennett [11]). The sequences \chi_H and \chi_K are strongly deep.

We conclude this section by noting the set of strongly deep sequences is "small," both in the sense of Lebesgue measure and in the sense of Baire category.

**Theorem 3.13.** The set strDEEP is meager and has measure 0. In fact, if t and c are as in Theorem 3.2, then D^t_{\epsilon} is meager and has measure 0.

**Proof.** Let t and c be as in Theorem 3.2. Then RAND \cap D^t_{\epsilon} = \emptyset. Since RAND has measure 1, it follows that D^t_{\epsilon} has measure 0.
To see that $D^f_e$ is meager, define a strategy $\tau : N \times \{0,1\}^* \rightarrow \{0,1\}^*$ for Player II in Banach-Mazur game $G[D^f_e]$ by

$$\tau_m(w) = w^{0\text{time}_v(\pi_w)},$$

where $\pi_w$ is a minimal program for $w$. An argument similar to the proof of Theorem 3.2 shows that $\tau$ is a winning strategy for Player II in $G[D^f_e]$.

$\Box$

3.2 Weak Computational Depth

In this section we examine Bennett's notion of weak computational depth. Stated simply, a sequence is weakly deep if it is not truth-table reducible to any algorithmically random sequence.

**Definition** (Bennett [11]). A sequence $x \in \{0,1\}^\infty$ is *weakly deep*, and we write $x \in \text{wkDEEP}$, if there is no sequence $z \in \text{RAND}$ such that $x \leq_u z$.

As noted by Bennett [11], this notion has been investigated in other forms by Levin and V'yugin [92, 50, 88, 54, 89, 53].

Here we use the notation

$$\text{REC}_{ut}(\text{RAND}) = \{x \in \{0,1\}^\infty \mid (\exists z \in \text{RAND}) x \leq_{ut} z\}.$$  

It is clear that every recursive and every algorithmically random sequence is in $\text{REC}_{ut}(\text{RAND})$. Moreover,

$$\text{wkDEEP} = \text{REC}_{ut}(\text{RAND})^c,$$
so no weakly deep sequence is recursive or algorithmically random. We now show that
every strongly deep sequence is weakly deep.

**Theorem 3.14** (Bennett [11]). \( \text{strDEEP} \subseteq \text{wkDEEP} \).

**Proof.** Assume that \( x \in \text{strDEEP} \) and \( x \leq_{tt} y \). To see that \( x \in \text{wkDEEP} \), it suffices
to show that \( y \not\in \text{RAND} \). But this follows immediately from Theorems 3.2 and 3.6.

Next we mention, but do not prove, that \( \text{wkDEEP} \) is "large" in the sense of
Baire Category. (See [37] for the proof of Theorem 3.15.)

**Theorem 3.15.** The set \( \text{wkDEEP} \) is comeager.

It follows immediately that, in the sense of Baire Category, almost every sequence
is weakly deep but not strongly deep.

**Corollary 3.16.** The set \( \text{wkDEEP} - \text{strDEEP} \) is comeager.

**Proof.** This follows immediately from Theorems 3.13 and 3.15.

Thus, as Bennett notes, there are weakly deep sequences that are not strongly
depth.

**Corollary 3.17** (Bennett [11]). \( \text{strDEEP} \not\subseteq \text{wkDEEP} \).

**Proof.** This follows immediately from Theorem 3.14 and Corollary 3.16.

Figure 3.2 summarizes the relationships among \( \text{REC}, \text{RAND}, \text{wkDEEP}, \) and
\( \text{strDEEP} \). In the sense of Lebesgue measure, almost every binary sequence is in
RAND. On the other hand, in the sense of Baire category, almost every binary sequence is in \( \text{wkDEEP} - \text{strDEEP} \).

Figure 3.2: A classification of binary sequences. RAND has measure 1, while \( \text{wkDEEP} - \text{strDEEP} \) is comeager.
CHAPTER 4. THE STRUCTURE OF EXPONENTIAL TIME

In this chapter we investigate the measure-theoretic structure of the exponential-time complexity classes \( E = \text{DTIME}(2^{\text{linear}}) \) and \( E_2 = \text{DTIME}(2^{\text{polynomial}}) \). This investigation was initiated by Lutz [60, 61]. The results of this chapter were arrived at jointly with Jack Lutz [38].

Here we investigate the structure of \( E \) and \( E_2 \) with respect to a single measure of uniform complexity: the "size" of complexity cores. Briefly, a complexity core is a set of uniformly "hard" instances. This notion was introduced by Lynch [69] and has been used in a number of contexts [24, 26, 76, 77, 16, 36, 81, 17, 25, 91, etc.]. Here we examine upper and lower bounds on the size of complexity cores for languages in \( E \) and \( E_2 \).

In section 4.1 we prove an almost everywhere lower bound on the size of complexity cores for languages in \( E \) and \( E_2 \). Specifically, we show that almost every language in \( E \) and \( E_2 \) has \( \{0,1\}^* \) as a \( \text{DTIME}(2^m) \)-complexity core. Intuitively, this result says that most languages in \( E \) and \( E_2 \) require \( 2^m \) steps to decide the membership question on almost every string in \( \{0,1\}^* \).

In section 4.2 we show a tight lower bound on the size of complexity cores for weakly \( \leq_m^p \)-hard languages for \( E \) and \( E_2 \). Specifically, we show that every weakly \( \leq_m^p \)-hard language for \( E \) or \( E_2 \) has a dense exponential complexity core. This result
extends work of Orponen and Schöning [77].

The results of sections 4.1 and 4.2 provide the basis for an investigation of the distribution of languages in $E$ and $E_2$. In section 4.3 we examine the distribution of languages in the polynomial-time many-one degrees in $E$ and $E_2$. The main result of this section is the Small Span Theorem. The Small Span Theorem says that for every language $A$ in $E$ either its lower $\leq_P$-span or its upper $\leq_m$-span has measure 0 in $E$ or $E_2$. The Small Span Theorem immediately implies that every $\leq_m$-degree has measure 0 in $E$ and in $E_2$. Thus the set of $\leq_P$-complete languages for NP, PSPACE, $E$, $E_2$, or any class whatsoever has measure 0 in $E$ and in $E_2$.

In section 4.4 we conclude this chapter by showing a tight upper bound on the size of complexity cores for the $\leq_P$-hard problems for $E$ and $E_2$. The main result of this section is that every $\text{DTIME}(2^{n^a})$-complexity core of every $\leq_P$-hard language for $E$ has a dense complement. Since this upper bound is much lower than the almost everywhere lower-bound of section 4.1, we say that $\leq_P$-complete problems for $E$ and $E_2$ are unusually simple.

4.1 The Distribution of Uniform Complexity

We begin our examination of the structure of exponential time by investigating the distribution of very complex languages in $E$ and $E_2$. Here we measure complexity in terms of the "size" of complexity cores. Stated simply, a complexity core is a set of uniformly hard instances. The main result of this section shows that almost every language in $E$ or $E_2$ has $\{0,1\}^*$ as a $\text{DTIME}(2^{\text{m}})$ complexity core. We first introduce some necessary notation.

Given a machine $M$ and an input $x \in \{0,1\}^*$, we write $M(x) = 1$ if $M$ accepts
x, $M(x) = 0$ if $M$ rejects $x$, and $M(x) = \bot$ in any other case (i.e., if $M$ fails to halt or $M$ halts without deciding $x$). If $M(x) \in \{0, 1\}$, we write $\text{time}_M(x)$ for the number of steps used in the computation of $M(x)$. If $M(x) = \bot$, we define $\text{time}_M(x) = \infty$.

We partially order the set $\{0, 1, \bot\}$ by $\bot < 0$ and $\bot < 1$, with 0 and 1 incomparable.

A machine $M$ is consistent with a language $A \subseteq \{0, 1\}^*$ if $M(x) \leq \|x \in A\|$ for all $x \in \{0, 1\}^*$.

**Definition.** Let $t : \mathbb{N} \to \mathbb{N}$ be a time bound and let $A, K \subseteq \{0, 1\}^*$. Then $K$ is a DTIME$(t(n))$-complexity core of $A$ if, for every $c \in \mathbb{N}$ and every machine $M$ that is consistent with $A$, the "fast set"

$$F = \{x \mid \text{time}_M(x) \leq c \cdot t(|x|) + c\}$$

satisfies $|F \cap K| < \infty$. (By our definition of $\text{time}_M(x)$, $M(x) \in \{0, 1\}$ for all $x \in F$.

Thus $F$ is the set of all strings that $M$ "decides efficiently.")

**Definition.** Let $A, K \subseteq \{0, 1\}^*$.

1. $K$ is a polynomial complexity core (or, briefly, a $P$-complexity core) of $A$ if $K$ is a DTIME$(n^k)$-complexity core of $A$ for all $k \in \mathbb{N}$.

2. $K$ is an exponential complexity core of $A$ if there is a real number $\epsilon > 0$ such that $K$ is a DTIME$(2^n)$-complexity core of $A$.

There is a close connection between complexity cores and a notion of "incompressibility" by many-one reductions. The notion of incompressibility was introduced by Meyer [74]. Here we define incompressibility in terms of collision sets.

**Definition.** The collision set of a function $f : \{0, 1\}^* \to \{0, 1\}^*$ is

$$C_f = \{x \in \{0, 1\}^* \mid (\exists y < x)f(y) = f(x)\}.$$
Here, we are using the standard ordering \( s_0 < s_1 < s_2 < \cdots \) of \( \{0,1\}^* \).

Note that \( f \) is one-to-one if and only if \( C_f = \emptyset \).

**Definition.** A function \( f : \{0,1\}^* \to \{0,1\}^* \) is **one-to-one almost everywhere** (or, briefly, **one-to-one a.e.**) if its collision set \( C_f \) is finite.

We say that a language \( A \) is **incompressible** if every function that is a reduction of \( A \) is one-to-one almost everywhere.

**Definition.** Let \( A, B \subseteq \{0,1\}^* \) and let \( t : \mathbb{N} \to \mathbb{N} \). A \( \leq_{\text{DTIME}(t)} \)-reduction of \( A \) to \( B \) is a function \( f \in \text{DTIME}(t) \) such that \( A = f^{-1}(B) \), i.e., such that, for all \( x \in \{0,1\}^* \), \( x \in A \) iff \( f(x) \in B \). A \( \leq_{\text{DTIME}(t)} \)-reduction of \( A \) is a function \( f \) that is a \( \leq_{\text{DTIME}(t)} \)-reduction of \( A \) to \( f(A) \).

**Definition.** Let \( t : \mathbb{N} \to \mathbb{N} \). A language \( A \subseteq \{0,1\}^* \) is **incompressible by** \( \leq_{\text{DTIME}(t)} \)-reductions if every \( \leq_{\text{DTIME}(t)} \)-reduction of \( A \) is one-to-one a.e. A language \( A \subseteq \{0,1\}^* \) is **incompressible by** \( \leq_{p} \)-reductions if it is incompressible by \( \leq_{\text{DTIME}(t)} \)-reductions for all polynomials \( q \).

As we mention above, there is a close connection between incompressibility and complexity cores. The following lemma shows that every incompressible language has a large complexity core.

**Lemma 4.1.** If \( t : \mathbb{N} \to \mathbb{N} \) is time constructible then every language that is incompressible by \( \leq_{\text{DTIME}(t)} \)-reductions has \( \{0,1\}^* \) as a \( \text{DTIME}(t) \)-complexity core.

**Proof.** Let \( A \) be a language that does not have \( \{0,1\}^* \) as a \( \text{DTIME}(t) \)-complexity core. It suffices to prove that \( A \) is not incompressible by \( \leq_{\text{DTIME}(t)} \)-reductions. This
is clear if $A = \emptyset$ or $A = \{0,1\}^*$, so assume that $\emptyset \neq A \neq \{0,1\}^*$. Fix $u \in A$ and $v \in A^c$. Since $\{0,1\}^*$ is not a \text{DTIME}(t)$-complexity core of $A$, there exist $c \in \mathbb{N}$ and a machine $M$ such that $M$ is consistent with $A$ and the fast set

$$F = \{x \mid \text{time}_M(x) \leq c \cdot t(|x|) + c\}$$

is infinite. Define a function $f : \{0,1\}^* \to \{0,1\}^*$ by

$$f(x) = \begin{cases} u & \text{if } M(x) = 1 \text{ in } \leq c \cdot t(|x|) + c \text{ steps} \\
v & \text{if } M(x) = 0 \text{ in } \leq c \cdot t(|x|) + c \text{ steps} \\
x & \text{otherwise.} \end{cases}$$

Since $t$ is time-constructible, $f \in \text{DTIME}(t)$. Since $M$ is consistent with $A$, $f$ is a $\leq_{m}^{\text{DTIME}(t)}$-reduction of $A$ to $A$. Since $F$ is infinite, at least one of the sets $f^{-1}\{u\}$, $f^{-1}\{v\}$ is infinite, so the collision set $C_f$ is infinite. Thus $A$ is not incompressible by $\leq_{m}^{\text{DTIME}(t)}$-reductions.

\begin{corollary}
Let $c \in \mathbb{N}$.
\begin{enumerate}
\item (Balcazar and Schöning [9]) Every language that is incompressible by $\leq_{m}^{P}$-reductions has $\{0,1\}^*$ as a $P$-complexity core.
\item Every language that is incompressible by $\leq_{m}^{\text{DTIME}(2^m)}$-reductions has $\{0,1\}^*$ as a $\text{DTIME}(2^m)$-complexity core.
\item Every language that is incompressible by $\leq_{m}^{\text{DTIME}(2^n)}$-reductions has $\{0,1\}^*$ as a $\text{DTIME}(2^n)$-complexity core.
\end{enumerate}
\end{corollary}

In order to show the main result of this section, we first prove that almost every language in $E$ or $E_2$ is incompressible by many-one reductions.
Theorem 4.3. Let $c \in \mathbb{Z}^+$ and define the sets

$$X = \{ A \subseteq \{0,1\}^* | \text{A is incompressible by } \leq_{m}^{\text{DTIME}(2^n)} \text{-reductions} \},$$

$$Y = \{ A \subseteq \{0,1\}^* | \text{A is incompressible by } \leq_{m}^{\text{DTIME}(2^n)} \text{-reductions} \}.$$ 

Then $\mu_p(X) = \mu_p(Y) = 1$. Thus almost every language in $E$ is incompressible by $\leq_{m}^{\text{DTIME}(2^n)}$-reductions, and almost every language in $E_2$ is incompressible by $\leq_{m}^{\text{DTIME}(2^n)}$-reductions.

Proof. Let $c \in \mathbb{Z}^+$. We prove that $\mu_p(X) = 1$. The proof that $\mu_p(Y) = 1$ is analogous.

Let $f \in \text{DTIME}(2^{(c+1)n})$ be a function that is universal for $\text{DTIME}(2^n)$, in the sense that

$$\text{DTIME}(2^n) = \{ f_i | i \in \mathbb{N} \}.$$ 

For each $i \in \mathbb{N}$, define a set $Z_i$ of languages as follows: If the collision set $C_{f_i}$ is finite, then $Z_i = \emptyset$. Otherwise, if $C_{f_i}$ is infinite, then $Z_i$ is the set of all languages $A$ such that $f_i$ is a $\leq_{m}^{\text{DTIME}(2^n)}$-reduction of $A$.

Define a function $d : \mathbb{N} \times \mathbb{N} \times \{0,1\}^* \rightarrow [0, \infty)$ as follows: Let $i, k \in \mathbb{N}$ be arbitrary, let $w \in \{0,1\}^*$, and let $b \in \{0,1\}$.

(i) $d_{i,k}(\lambda) = 2^{-k}$.

(ii) If $s_{|w|} \not\in C_{f_i}$, then $d_{i,k}(wb) = d_{i,k}(w)$.

(iii) If $s_{|w|} \in C_{f_i}$, then fix the least $j \in \mathbb{N}$ such that $f_i(s_j) = f_i(s_{|w|})$ and set

$$d_{i,k}(wb) = 2 \cdot d_{i,k}(w) \cdot [b = w[j]].$$
It is clear that $d$ is a 2-DS. Since $f \in \text{DTIMEF}(2^{c+1}n)$ and the computation of $d_{i,k}(w)$ only uses values $f_i(u)$ for strings $u$ with $|u| = O(|w|)$, it is also clear that $d \in p$, so $d$ is a p-computable 2-DS.

We now show that $Z_i \subseteq S[d_{i,k}]$ for all $i, k \in \mathbb{N}$. If $C_{f_i}$ is finite, then this is clear (because $Z_i = \emptyset$), so assume that $C_{f_i}$ is infinite and let $A \in Z_i$. Let $w$ be a string consisting of the first $l$ bits of the characteristic sequence of $A$, where $s_l$ is the $k$th element of $C_{f_i}$. This choice of $l$ ensures that clause (iii) of the definition of $d$ is invoked exactly $k$ times in the recursive computation of $d_{i,k}(w)$. Since $f_i$ is a $\leq_{m}^{\text{DTIME}(2^{cn})}$-reduction of $A$ (because $A \in Z_i$), we have $b = w[j]$ in each of these $k$ invocations, so

$$d_{i,k}(w) = 2^k \cdot d_{i,k}(\lambda) = 1.$$ 

Thus $A \in C_w \subseteq S[d_{i,k}]$. This confirms that $Z_i \subseteq S[d_{i,k}]$ for all $i, k \in \mathbb{N}$. It follows easily that, for each $i \in \mathbb{N}$, $d_i$ is a p-null cover of $Z_i$. This implies that

$$X^c = \bigcup_{k=0}^{\infty} Z_k$$

is a p-union of p-measure 0 sets, whence $\mu_p(X) = 1$ by Lemma 2.1. 

**Corollary 4.4.** Almost every language in $E$ and almost every language in $E_2$ is incompressible by $\leq^p_m$-reductions. 

**Corollary 4.5** (Meyer[74]). There is a language $A \in E$ that is incompressible by $\leq^p_m$-reductions. 

Theorem 4.3 immediately implies that almost every language in $E$ or $E_2$ has very large complexity cores.
Corollary 4.6. Let $c \in \mathbb{Z}^+$.

1. Almost every language in $E$ has $\{0,1\}^*$ as a $\text{DTIME}(2^{cn})$-complexity core.
2. Almost every language in $E_2$ has $\{0,1\}^*$ as a $\text{DTIME}(2^{cn})$-complexity core. □

4.2 The Complexity of Weakly Hard Problems: Lower Bounds

In this section we prove a tight lower bound on the size of complexity core for weakly $\leq_p$-hard problems for $E$. The results of this section extend the following two lower bound results of Orponen and Schöning [77].

Fact 4.7 (Orponen and Schöning [77]). Every language that is $\leq_p$-hard for $E$ (equivalently, for $E_2$) has a dense $P$-complexity core.

Fact 4.8 (Orponen and Schöning [77]). If $P \neq NP$, then every language that is $\leq_p$-hard for $NP$ has a nonsparse $P$-complexity core.

Our first result shows that every weakly $\leq_p$-hard language for $E$ or $E_2$ has a dense exponential complexity core. (Recall that a language is weakly $\leq_p$-hard for $E$ if the set $P_m(A) = \{B \mid B \leq_p A\}$ does not have measure 0 in $E$.)

Theorem 4.9. Every language that is weakly $\leq_p$-hard for $E$ or $E_2$ has a dense exponential complexity core.

Proof. We prove this for $E$. The proof for $E_2$ is identical.

Let $H$ be a language that is weakly $\leq_p$-hard for $E$. Then $P_m(H)$ does not have measure 0 in $E$, so by Theorem 4.3, there is a language $A \in P_m(H)$ that is incompressible by $\leq_D^{\text{DTIME}(2^n)}$-reductions. Let $f$ be a $\leq_m$-reduction of $A$ to $H$, let $q$ be a strictly increasing polynomial bound on the time required to compute $f$, and let...
\[ e = \frac{1}{3^{\deg(e)}}. \] Then the language \( K = f(\{0,1\}^*) \) is a dense \( \text{DTIME}(2^{n^e}) \)-complexity core of \( H \).

Since Lutz [65] has shown the existence of weakly \( \leq_p \)-complete languages for \( E \) that are not \( \leq_p \)-complete for \( E \), Theorem 4.9 extends Fact 4.7. Furthermore, if \( \text{NP} \) is "not small" in \( E \) or \( E_2 \), then Theorem 4.9 implies that every \( \text{NP} \)-hard problem has a dense exponential complexity core.

**Corollary 4.10.** If \( \mu(\text{NP} \mid E) \neq 0 \) or \( \mu(\text{NP} \mid E_2) \neq 0 \), then every \( \leq_p \)-hard language for \( \text{NP} \) has a dense exponential complexity core.

Notice that Corollary 4.10 obtains a stronger consequence than Fact 4.8 at the expense of a stronger hypothesis.

We conclude this section by mentioning that Theorem 4.9 is tight, even when we restrict our attention to problems that are \( \leq_p \)-complete for \( \text{NP} \), \( E \), or \( E_2 \).

**Theorem 4.11.** For every \( \epsilon > 0 \), each of the classes \( \text{NP} \), \( E \), and \( E_2 \) has a \( \leq_p \)-complete language, every \( \text{P} \)-complexity core \( K \) of which satisfies \( |K_{\leq_p}| < 2^{n^e} \) a.e.

**Proof.** Let \( \epsilon > 0 \), let \( C \) be any one of the classes \( \text{NP} \), \( E \), \( E_2 \), and let \( A \) be a language that is \( \leq_m \)-complete for \( C \). Let \( k = \lceil \frac{1}{\epsilon} \rceil \) and define the language

\[ B = \{ x0^{\lfloor |x|/k \rfloor} \mid x \in A \}. \]

Then \( B \) is \( \leq_p \)-complete for \( C \) and every \( \text{P} \)-complexity core \( K \) of \( B \) satisfies \( |K_{\leq_p}| < 2^{n^e} \) a.e. \( \Box \)
4.3 The Measure of Degrees

In this section we examine the measure of the $\leq^P_m$-degrees in $E$ and $E_2$. We show that every $\leq^P_m$-degree has measure 0 in $E$. This result implies, among other things, that the set of NP-complete problems has measure 0 in $E$ and is a measure 0 subset of $E_2$. The fact that every $\leq^P_m$-degree has measure 0 in $E$ is seen here as a consequence of a more general phenomenon, namely, that for every language $A$ either its upper $\leq^P_m$-span or its lower $\leq^P_m$-span has measure 0 in $E$. We refer to this result as the Small Span Theorem and prove it first. We first recall some notation.

Recall from Chapter 2 that the lower $\leq^P_m$-span of a language $A \subseteq \{0,1\}^\omega$ is

$$P_m(A) = \{B \subseteq \{0,1\}^\ast \mid B \leq^P_m A\}$$

and the upper $\leq^P_m$-span of $A$ is

$$P_m^{-1}(A) = \{B \subseteq \{0,1\}^\ast \mid A \leq^P_m B\}.$$

The $\leq^P_m$-degree of $A$ is

$$\deg_{m}^P(A) = P_m(A) \cap P_m^{-1}(A).$$

We now show that, for every language $A \subseteq \{0,1\}^\ast$, either $P_m(A)$ or $P_m^{-1}(A)$ is "small" in $E$.

**Theorem 4.12** (Small Span Theorem).

1. For every $A \in E$,

$$\mu(P_m(A) \mid E) = 0$$

or

$$\mu_p(P_m^{-1}(A)) = \mu(P_m^{-1}(A) \mid E) = 0.$$
2. For every $A \in E_2$, 
\[ \mu(P_m(A) \mid E_2) = 0 \]
or 
\[ \mu_{p_2}(P_{m}^{-1}(A)) = \mu(P_m^{-1}(A) \mid E_2) = 0. \]

We prove the Small Span Theorem using the following technical lemma. We then prove the lemma.

**Lemma 4.13.** Let $A$ be a language that is incompressible by $\leq_P$-reductions.

1. If $A \in E$, then $\mu(P_m^{-1}(A)) = \mu(P_m^{-1}(A) \mid E) = 0$.
2. If $A \in E_2$, then $\mu_{p_2}(P_m^{-1}(A)) = \mu(P_m^{-1}(A) \mid E_2) = 0$.

**Proof of Theorem 4.12.**

To prove 1, let $A \in E$ and let $X$ be the set of all languages that are incompressible by $\leq_P$-reductions. We have two cases.

**Case I.** If $P_m(A) \cap E \cap X = \emptyset$, then Corollary 4.4 tells us that $\mu(P_m(A) \mid E) = 0$.

**Case II.** If $P_m(A) \cap E \cap X \neq \emptyset$, then fix a language $B \in P_m(A) \cap E \cap X$.

Since $B \in E \cap X$, Lemma 4.13 tells us that 
\[ \mu_p(P_m^{-1}(B)) = \mu(P_m^{-1}(B) \mid E) = 0. \]

Since $P_m^{-1}(A) \subseteq P_m^{-1}(B)$, it follows that 
\[ \mu_p(P_m^{-1}(A)) = \mu(P_m^{-1}(A) \mid E) = 0. \]

This proves 1. The proof of 2 is identical. \qed
Proof of Lemma 4.13.

To prove 1, let $A \in E$ be incompressible by $\leq^P_m$-reductions. Let $f \in \text{DTIMEF}(2^n)$ be a function that is universal for PF, in the sense that

$$\text{PF} = \{ f_i \mid i \in \mathbb{N} \}.$$ 

For each $i \in \mathbb{N}$, define the set $Z_i$ of languages as follows. If the collision set $C_{f_i}$ is infinite, then $Z_i = \emptyset$. Otherwise, if $C_{f_i}$ is finite, then

$$Z_i = \{ B \subseteq \{0,1\}^* \mid A \leq^P_m B \text{ via } f_i \}.$$

Note that

$$P^{-1}_m(A) = \bigcup_{i=0}^{\infty} Z_i$$

because $A$ is incompressible by $\leq^P_m$-reductions.

Define a function $d : \mathbb{N} \times \mathbb{N} \times \{0,1\}^* \to [0, \infty)$ as follows. Let $i, k \in \mathbb{N}$ be arbitrary, let $w \in \{0,1\}^*$, and let $b \in \{0,1\}$.

(i) $d_{i,k}(\lambda) = 2^{-k}$.

(ii) If there is no $j \leq 2|w|$ such that $f_i(s_j) = s_{|w|}$, then $d_{i,k}(wb) = d_{i,k}(w)$.

(iii) If there exists $j \leq 2|w|$ such that $f_i(s_j) = s_{|w|}$, then fix the least such $j$ and set

$$d_{i,k}(wb) = 2 \cdot d_{i,k}(w) \cdot [b = [s_j \in A]].$$

It is clear that $d$ is a 2-DS. Also, since $f \in \text{DTIMEF}(2^n)$ and $A \in E$, it is easy to see that $d \in p$, whence $d$ is a $p$-computable 2-DS.

We now show that $Z_i \subseteq S[d_{i,k}]$ for all $i, k \in \mathbb{N}$. If $C_{f_i}$ is infinite, then this is clear (because $Z_i = \emptyset$), so assume that $|C_{f_i}| = c < \infty$ and let $B \in Z_i$, i.e., $A \leq^P_m B$.
via \( f_i \). Let \( v \) be the string consisting of the first \( l \) bits of the characteristic sequence of \( B \), where \( l \) is large enough that

\[
    f_i(\{s_0, ..., s_{2k+4c-1}\}) \subseteq \{s_0, ..., s_{l-1}\}.
\]

Consider the computation of \( d_{i,k}(v) \) by clauses (i), (ii), and (iii) above. Since \( A \preceq_m B \) via \( f_i \), clause (iii) does not cause \( d_{i,k}(w) \) to be 0 for any prefix \( w \) of \( v \). Let

\[
    S = \{s_n \mid 0 \leq n < 2k + 4c \text{ and } f_i(s_n) \notin \{s_0, ..., s_{[\frac{l}{2}] - 1}\}\}
\]

and

\[
    T = f_i(S).
\]

Then clause (iii) doubles the density whenever \( s_{|w|} \in T \), so

\[
    d_{i,k}(v) \geq 2^{|T|}d_{i,k}(\lambda) = 2^{|T| - k} \geq 2^{|S| - k - c}.
\]

Also, if

\[
    S' = \{s_n \mid 0 \leq n < 2k + 4c \text{ and } f_i(s_n) \notin \{s_0, ..., s_{k+2c-1}\}\},
\]

then \( S' \subseteq S \) and

\[
    |S'| \geq (2k + 4c) - (k + 2c) - c = k + c.
\]

Putting this all together, we have

\[
    d_{i,k}(v) \geq 2^{|S| - k - c} \geq 2^{|S'| - k - c} \geq 1,
\]

whence \( B \in C_v \subseteq S[d_{i,k}] \). This shows that \( Z_i \subseteq S[d_{i,k}] \) for all \( i, k \in \mathbb{N} \).

Since \( d \) is \( p \)-computable and \( d_{i,k}(\lambda) = 2^{-k} \) for all \( i, k \in \mathbb{N} \), it follows that, for all \( i \in \mathbb{N} \), \( d_i \) is \( p \)-null cover of \( Z_i \). This implies that \( P_m^{-1}(A) \) is a \( p \)-union of the
p-measure 0 sets \( Z_i \). It follows by Lemma 2.1 that \( \mu_p(P_m^{-1}(A)) = \mu(P_m^{-1}(A) \mid E) = 0 \). This completes the proof of 1.

The proof of 2 is identical. One need only note that, if \( A \in E_2 \), then \( d \in p_2 \). □

Note that Ambos-Spies [5] has shown that the set \( P_m^{-1}(A) \) has Lebesgue measure 0 whenever \( A \notin \mathbb{P} \). However, this result says nothing about the "size" of \( P_m^{-1}(A) \) in \( E \) or \( E_2 \) since both of these classes also have Lebesgue measure 0. Lemma 4.13 obtains a similar result, namely, that \( P_m^{-1}(A) \) has p-measure 0 from a stronger hypothesis on \( A \).

Lemma 4.13 immediately implies that the set of \( \leq^p_m \)-hard languages for \( E \) has p-measure 0.

**Theorem 4.14.** Let \( \mathcal{H}_E \) be the set of all languages that are \( \leq^p_m \)-hard for \( E \). Then \( \mu_p(\mathcal{H}_E) = 0 \).

**Proof.** Let \( A \) be as in Corollary 4.5. Then \( \mathcal{H}_E \subseteq P_m^{-1}(A) \), so Lemma 3.1 tells us that

\[
\mu_p(\mathcal{H}_E) = \mu_p(P_m^{-1}(A)) = 0.
\]

□

Theorem 4.14 allows us to conclude that \( \leq^p_m \)-complete problems are rare in \( E \) and \( E_2 \), a result proven by Mayordomo [72] using other techniques.

**Corollary 4.15** (Mayordomo[72]). Let \( C_E, C_{E_2} \) be the sets of languages that are \( \leq^p_m \)-complete for \( E, E_2 \), respectively. Then \( \mu(C_E \mid E) = \mu(C_{E_2} \mid E_2) = 0 \). □

(Mayordomo's proof of Corollary 4.15 uses the fact, proven by Berman [13], that every \( \leq^p_m \)-complete problem has an infinite polynomial-time decidable subset.)
Corollary 4.15 is seen here as a special case of a more general phenomenon, namely, that every $\leq_m^P$-degree has measure 0 in $E$ and $E_2$.

**Theorem 4.16.** For all $A \subseteq \{0,1\}^*$,

$$
\mu(\deg_m^P(A) \mid E) = \mu(\deg_m^P(A) \mid E_2) = 0.
$$

**Proof.** Let $A \subseteq \{0,1\}^*$. We prove that $\mu(\deg_m^P(A) \mid E) = 0$. The proof that $\mu(\deg_m^P(A) \mid E_2) = 0$ is identical (in fact simpler, because $E_2$ is closed under $\leq_m^P$).

If $\deg_m^P(A) \cap E = \emptyset$, then $\mu(\deg_m^P(A) \mid E) = 0$ holds trivially, so assume that $\deg_m^P(A) \cap E \neq \emptyset$. Fix $B \in \deg_m^P(A) \cap E$. Then, by Theorem 4.12,

$$
\mu(\deg_m^P(B) \mid E) = \mu(P_m(B) \mid E) = 0
$$
or

$$
\mu(\deg_m^P(B) \mid E) = \mu(P_m^{-1}(B) \mid E) = 0.
$$

Since $\deg_m^P(A) = \deg_m^P(B)$, it follows that $\mu(\deg_m^P(A) \mid E) = 0$. \qed

We now mention two corollaries for NP. The first says that the set of NP-hard problems (i.e., languages that are $\leq_m^P$-hard for NP) has $p$-measure 0 if NP is “not small” in $E$. The second says that the set of NP-complete problems is “small” in both $E$ and $E_2$.

**Corollary 4.17.** Let $\mathcal{H}_{NP}$ be the set of languages that are $\leq_m^P$-hard for NP.

1. If $\mu(NP \mid E) \neq 0$, then $\mu_p(\mathcal{H}_{NP}) = \mu(\mathcal{H}_{NP} \mid E) = 0$.
2. If $\mu(NP \mid E_2) \neq 0$, then $\mu_{p_2}(\mathcal{H}_{NP}) = \mu(\mathcal{H}_{NP} \mid E_2) = 0$.

**Proof.** This follows immediately from Theorem 4.12, with $A = SAT$. \qed
Corollary 4.18. Let $C_{NP}$ be the set of languages that are $\leq_P^m$-complete for NP. Then $\mu(C_{NP} \mid E) = \mu(C_{NP} \mid E_2) = 0$.

Proof. Since $C_{NP} = \text{deg}_{m}^P(\text{SAT})$, this follows immediately from Theorem 4.16.

4.4 The Complexity of Hard Problems: Upper Bounds

In this section we show a tight upper bound on the size of complexity cores for $\leq_P^m$-hard problems for E. The upper bound implies that the $\leq_P^m$-complete complete problems for E are unusually simple in the sense that they have smaller complexity cores than almost every language in E. We first show that every $\leq_P^m$-hard language for E is DTIME($2^{4n}$) decidable on a dense, DTIME($2^{4n}$) decidable set of inputs.

Theorem 4.19. For every $\leq_P^m$-hard language $H$ for E, there exist $B, D \in \text{DTIME}(2^{4n})$ such that $D$ is dense and $B = H \cap D$.

Proof. By Corollary 4.5, there is a language in E that is incompressible by $\leq_P^m$-reductions. In fact, Meyer’s construction [74] shows that there is a language $A \in \text{DTIME}(5^n)$ that is incompressible by $\leq_P^m$-reductions. As in Fact 4.7 and Theorem 4.9, this idea has often been used to establish lower bounds on the complexities of $\leq_P^m$-hard languages. Here we use it to establish an upper bound.

The following simple notation is useful here. The nonreduced image of a language $S \subseteq \{0,1\}^*$ under a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is

$$f^2(S) = \{f(x) \mid x \in S \text{ and } |f(x)| \geq |x|\}.$$ 

Note that

$$f^2(f^{-1}(S)) = S \cap f^2(\{0,1\}^*)$$
for all $f$ and $S$.

Let $H$ be $\leq^p$-hard for $E$. Then there is a $\leq^p$-reduction $f$ of $A$ to $H$. Let $B = f^\geq(A), D = f^\geq(\{0,1\}^*)$. Since $A \in \text{DTIME}(5^n)$ and $f \in \text{PF}$, it is clear that $B, D \in \text{DTIME}(10^n) \subseteq \text{DTIME}(2^{4n})$.

Fix a polynomial $q$ and a real number $\epsilon > 0$ such that $|f(x)| \leq q(|x|)$ for all $x \in \{0,1\}^*$ and $q(n^{2\epsilon}) < n$ a.e. Let $W = \{x \mid |f(x)| < |x|\}$. Then, for all sufficiently large $n \in \mathbb{N}$, writing $m = \lceil n^{2\epsilon} \rceil$, we have

\[
f(\{0,1\}^{\leq_m}) - \{0,1\}^{<m} \subseteq f(\{0,1\}^{\leq_m}) - f(W^{\leq_m})
\]
\[
\subseteq f^\geq(\{0,1\}^{\leq_m})
\]
\[
\subseteq D_{\leq q(m)}
\]
\[
\subseteq D_{\leq n},
\]

whence

\[
|D_{\leq n}| \geq |f(\{0,1\}^{\leq_m})| - |\{0,1\}^{<m}|
\]
\[
\geq |\{0,1\}^{\leq_m}| - |C_f| - |\{0,1\}^{<m}|
\]
\[
= 2^m - |C_f|.
\]

Since $|C_f| < \infty$, it follows that $|D_{\leq n}| > 2^n\epsilon$ for all sufficiently large $n$. Thus $D$ is dense.

Finally, note that $B = f^\geq(A) = f^\geq(f^{-1}(H)) = H \cap f^\geq(\{0,1\}^*) = H \cap D$. This completes the proof. \qed

Theorem 4.19 immediately implies that every $\text{DTIME}(2^{4n})$ complexity core of every $\leq^p_m$-hard language for $E$ must have a dense complement.
Theorem 4.20. Every \( \text{DTIME}(2^{4n}) \)-complexity core of every \( \leq_m^P \)-hard language for \( E \) has a dense complement.

**Proof.** Let \( H \) be \( \leq_m^P \)-hard for \( E \) and let \( K \) be a \( \text{DTIME}(2^{4n}) \)-complexity core of \( H \). Choose \( B, D \) for \( H \) as in Theorem 4.19. Fix machines \( M_B \) and \( M_D \) that decide \( B \) and \( D \), respectively, with \( \text{time}_{M_B}(x) = O(2^{4|w|}) \) and \( \text{time}_{M_D}(x) = O(2^{4|w|}) \). Let \( M \) be a machine that implements the following algorithm.

```
begin
    input x;
    if \( M_D(x) \) accepts
        then simulate \( M_B(x) \)
    else run forever
end M.
```

Then \( x \in D \Rightarrow M(x) = [x \in B] = [x \in H \cap D] = [x \in H] \) and \( x \notin D \Rightarrow M(x) = \bot \leq [x \in H] \), so \( M \) is consistent with \( H \). Also, there is a constant \( c \in \mathbb{N} \) such that for all \( x \in D \),

\[
\text{time}_M(x) \leq c \cdot 2^{4n} + c.
\]

Since \( K \) is a \( \text{DTIME}(2^{4n}) \)-complexity core of \( H \), it follows that \( K \cap D \) is finite. But \( D \) is dense, so this implies that \( D - K \) is dense, whence \( K^c \) is dense. \( \square \)

We conclude this chapter by showing that the upper bound of Theorem 4.20 is tight, even when we restrict our attention to the \( \leq_m^P \)-complete languages for \( E \) and \( E_2 \).
Theorem 4.21. Let $c \in \mathbb{N}$ and $0 < \epsilon \in \mathbb{R}$.

1. $E$ has a $\leq^\mathbb{P}_m$-complete language with a DTIME($2^{2^n}$)-complexity core $K$ that satisfies $|K_{\leq n}| > 2^{n+1} - 2^n$ a.e.

2. $E_2$ has a $\leq^\mathbb{P}_m$-complete language with a DTIME($2^{2^n}$)-complexity core $K$ that satisfies $|K_{\leq n}| > 2^{n+1} - 2^n$ a.e.

Proof. We prove the result for $E$. The proof for $E_2$ is similar.

Let $A$ be a language that is $\leq^\mathbb{P}_m$-complete for $E$ and let $k = \lfloor \frac{c}{2} \rfloor$. By Corollary 4.6, fix a language $B \in E$ that has $\{0,1\}^*$ as a DTIME($2^{2^n}$)-complexity core. Let

$$D = \{x10^{2^n} | x \in \{0,1\}^*\}$$

and define the languages

$$C = (B - D) \cup \{x10^{2^n} | x \in A\}$$

and

$$K = D^c.$$

It is clear that $C$ is $\leq^\mathbb{P}_m$-complete for $E$. Also, for all sufficiently large $n$,

$$|D_{\leq n}| = \sum_{m=0}^{n} |D_{=m}| \leq \sum_{m=0}^{n} 2^{m/4} \leq (n+1)2^{n/4} \leq (n+1)2^{n/4} < 2^n - 1,$$

so

$$|K_{\leq n}| = 2^{n+1} - 1 - |D_{\leq n}| > 2^{n+1} - 2^n$$ a.e.

We complete the proof by showing that $K$ is a DTIME($2^{2^n}$)-complexity core for $C$. For this, let $s \in \mathbb{N}$, let $M$ be a machine that is consistent with $C$, and define the fast set

$$F = \{x \mid time_M(x) \leq a \cdot 2^{2|x|} + a\}.$$
It suffices to prove that $|K \cap F| < \infty$.

Let $\hat{M}$ be a machine (designed in the obvious way) such that, for all $y \in \{0,1\}^*$,

$$\hat{M}(y) = \begin{cases} M(y) & \text{if } y \not\in D \\ \bot & \text{if } y \in D. \end{cases}$$

Then $\hat{M}$ is consistent with $B$ (because $B - D = C - D$ and $M$ is consistent with $C$) and $\{0,1\}^*$ is a DTIME($2^m$)-complexity core for $B$, so the fast set

$$\hat{F} = \{x \mid time_{\hat{M}}(x) \leq (a + 1)2^{|x|} + a\}$$

is finite. Since $K \cap F = F - D$ and $(F - D) - \hat{F}$ is finite, it follows that $|K \cap F| < \infty$, completing the proof. \qed
CHAPTER 5. THE STRUCTURE OF EXPONENTIAL SPACE

In Chapter 4 we investigate the uniform structure of $E$ and $E_2$. The results of Chapter 4 naturally extend to the exponential space classes $E^SPACE = DSPACE(2^{\text{linear}})$ and $E_2^SPACE = DSPACE(2^{\text{polynomial}})$. Here we instead investigate the nonuniform structure of these classes. This investigation was initiated by Lutz in [60, 61, 63]. The results of this section are joint work with Jack Lutz [39, 40].

Here we investigate the structure of $E^SPACE$ and $E_2^SPACE$ with respect to two measures of nonuniform complexity: space-bounded Kolmogorov complexity complexity and the “size” of nonuniform complexity cores. We examine both upper and lower bounds on the complexity of languages in $E^SPACE$ and $E_2^SPACE$ with respect to these two measures.

In section 5.1, we prove almost everywhere lower bounds on the space-bounded Kolmogorov complexity and size of nonuniform complexity cores for languages in $E^SPACE$ and $E_2^SPACE$. Specifically, we show that almost every language $A$ in $E^SPACE$ or $E_2^SPACE$ has space-bounded Kolmogorov complexity

$$K^{2^{en}}(A_{=n}) > 2^n - n \text{ a.e.}$$

This result extends the work of Lutz [63] and immediately implies that almost every language in $E^SPACE$ or $E_2^SPACE$ has $\{0,1\}^*$ as a $DSPACE(2^{en})/\text{Poly}$ complexity core.
In section 5.2, we show tight lower bounds on the space-bounded Kolmogorov complexity and size of nonuniform complexity cores for weakly $\leq_{m}^{p/poly}$-hard languages for ESPACE and $E_2$SPACE. These lower bounds extend the work of Huynh [34, 36].

In section 5.3 we show tight upper bounds on the complexity of $\leq_{m}^{p/poly}$-hard languages for ESPACE and $E_2$SPACE. Specifically, we show that every $\leq_{m}^{p/poly}$-hard language for ESPACE has space-bounded Kolmogorov complexity

$$KS^{23n}(A_m) < 2^n - 2^{n^c} \text{ i.o.}$$

Since this upper bound is much lower than the almost everywhere lower-bound of section 5.1, we say that the $\leq_{m}^{p/poly}$-complete languages for ESPACE are also unusually simple.

The results of sections 5.1, 5.2, and 5.3 provide the basis for an investigation of the distribution of hard languages in ESPACE and $E_2$SPACE. From the almost everywhere lower bounds of section 5.1 and the upper bounds of section 5.3 we conclude in Theorem 5.24 that the set of $\leq_{m}^{p/poly}$-complete problems form a measure 0 subset of ESPACE. In section 5.4 we show that Theorem 5.24 is a special case of a more general phenomenon, namely, that every $\leq_{m}^{p/poly}$-degree has measure 0 in ESPACE. This result implies that P/Poly has measure 0 in ESPACE, a result first proven by Lutz [63].

5.1 The Distribution of Nonuniform Complexity

In this section we investigate the distribution of languages that have high nonuniform complexity. We use space-bounded Kolmogorov complexity, nonuniform complexity cores and incompressibility by nonuniform reductions as measures of nonuni-
form complexity. The main results of this section show that almost every language in ESPACE is very complex with respect to each of these measures.

This section is organized as follows. In section 5.1.1 we investigate the distribution of languages with high space-bounded Kolmogorov complexity. Specifically, we prove that almost every language $A$ in ESPACE has space-bounded Kolmogorov complexity $K_{S^{2^n}}(A_{\leq n}) > 2^n - \sqrt{n}$ for almost every $n$. In section 5.1.2 we investigate the distribution of languages with large nonuniform complexity cores. More precisely, we prove that almost every language in ESPACE has $\{0,1\}^*$ as a $\text{DSPACE}(2^n)/\text{Poly}$ complexity core. Finally, in section 5.1.3 we investigate the distribution of languages that are incompressible by nonuniform many-one reductions. There we prove that almost every language in ESPACE is $n^{\log n}$-incompressible by $\leq^m_{\text{DSPACE}(2^n)/\text{poly}}$-reductions.

We begin by investigating the distribution of languages with high space-bounded Kolmogorov complexity.

5.1.1 Kolmogorov Complexity

The distribution of languages in ESPACE with high Kolmogorov complexity was first investigated in [63]. Here we strengthen the results of [63] in two important directions. First, we show that the almost everywhere lower bound of $2^{n+1} - 2^n$ on the space-bounded Kolmogorov complexity $K_{S^{2^n}}(A_{\leq n})$ is tight and cannot be improved (see Theorem 5.3). Next, we improve the almost everywhere lower bound on the space-bounded Kolmogorov complexity $K_{S^{2^n}}(A_{\leq n})$ from $2^n - 2^n$ to $2^n - n^c$ (Corollary 5.5). (Recall from Chapter 2 that $K_{S^{2^n}}(A_{\leq n})$ and $K_{S^{2^n}}(A_{\leq n})$ are the $2^n$-space bounded Kolmogorov complexities of $\chi_{A_{\leq n}}$ and $\chi_{A_{\leq n}}$, respectively.)
Theorem 5.1 (Lutz [63]). Let $c \in \mathbb{N}$ and $\epsilon > 0$.

(a) If
\[
X = \{A \subseteq \{0,1\}^* : KS^{2cn}(A_{\leq n}) > 2^n - 2^c \text{ a.e.}\},
\]
then $\mu_{\text{p-space}}(X) = \mu(X | \text{ESPACE}) = 1$.

(b) If
\[
Y = \{A \subseteq \{0,1\}^* : KS^{2cn}(A_{\leq n}) > 2^{n+1} - 2^c \text{ a.e.}\},
\]
then $\mu_{\text{p-space}}(Y) = \mu(Y | \text{ESPACE}) = 1$.

Although the lower bounds of Theorem 5.1 have been useful in a variety of applications (see [62, 63], for example), they are not strong enough for our purposes. For this reason, we ask the natural question: Can the almost everywhere lower bounds of Theorem 5.1 be improved?

We first consider Theorem 5.1(b). Martin-Löf [71] has shown that, for every real $a > 1$, almost every language $A \subseteq \{0,1\}^*$ has space-bounded Kolmogorov complexity
\[
KS^{2cn}(A_{\leq n}) > 2^{n+1} - an \text{ a.e.} \quad (5.1)
\]
(In fact, Martin-Löf showed that this holds even in the absence of a space bound.) The following known bounds show that the lower bound (5.1) is relatively tight.

Theorem 5.2. There exist constants $c_1, c_2 \in \mathbb{N}$ such that every language $A$ satisfies the following two conditions.

(i) $KS^{2n}(A_{\leq n}) < 2^{n+1} + c_1$ for all $n$.

(ii) $KS^{2cn}(A_{\leq n}) < 2^{n+1} - \log n + c_1$ i.o.
(Part (i) of Theorem 5.2 is well known and obvious. Part (ii) extends a result of Martin-Löf [71].)

**Theorem 5.3.** For every language \( A \in \text{ESPACE} \), there exists a real \( \varepsilon > 0 \) such that

\[
KS^{2^n}(A_{\leq n}) < 2^{n+1} - 2^n \text{ a.e.}
\]

**Proof.** Fix \( A \in \text{ESPACE} \) and \( a \in \mathbb{N} \) such that \( A \in \text{DSPACE}(2^{an}) \). For each \( n \in \mathbb{N} \), let \( n' = \left\lfloor \frac{n}{a+1} \right\rfloor \) and let \( y_n \) be the string of length \( 2^{n+1} - 2^{n'+1} \) such that \( \chi_{A_{\leq n}} = \chi_{A_{\leq n'}}y_n \).

Let \( M \) be a machine that, on input \((y,n)\), computes \( \chi_{A_{\leq n'}} \) using \( \leq 2^{an'} \) space and then outputs \( \chi_{A_{\leq n'}}y \). Let \( c \) be the optimality constant for the machine \( M \) (given by the definition of the optimal machine \( U \) at the beginning of this section). Then \( M(y_n,n) \) outputs \( \chi_{A_{\leq n}} \) in \( \leq 2^{an'} \) space, so for all sufficiently large \( n \), we have

\[
KS^{2^n}(A_{\leq n}) \leq KS^{2^{an'}}(A_{\leq n}) + c \\
\leq |y_n| + c \\
= 2^{n+1} - 2^{n'+1} + c \\
< 2^{n+1} - 2^n,
\]

where \( \varepsilon = \frac{1}{a+2} \). \( \square \)

Thus we cannot hope to improve Theorem 5.1(b).

An elementary counting argument shows that, for every \( c \in \mathbb{N} \), there exists a language \( A \in \text{ESPACE} \) with \( KS^{2^c}(A_{=n}) \geq 2^n \) for all \( n \in \mathbb{N} \). This suggests that the prospect for improving Theorem 5.1(a) may be more hopeful. In fact, we have the following almost everywhere lower bound result. (Recall the definition of \( p \)-convergence from section 2.4.1.)
Theorem 5.4. Let $c \in \mathbb{N}$ and let $f : \mathbb{N} \to \mathbb{N}$ be such that $f \in \text{pspace}$ and $\sum_{n=0}^{\infty} 2^{-f(n)}$ is p-convergent. If

$$X = \{ A \subseteq \{0,1\}^* \mid KS^{2^n}(A_{=n}) > 2^n - f(n) \text{ a.e.} \},$$

then $\mu_{\text{pspace}}(X) = \mu(X \mid \text{ESPACE}) = 1$.

Proof. Assume the hypothesis. By Lemma 2.2, it suffices to exhibit a pspace-computable 1-DS $d$ such that

$$\sum_{n=0}^{\infty} d_n(\lambda) \text{ is p-convergent} \quad (5.2)$$

and

$$X^c \subseteq \bigcap_{n=t}^{\infty} \bigcup_{n=t}^{\infty} S[d_n]. \quad (5.3)$$

Some notation will be helpful. For $n \in \mathbb{N}$, let

$$B_n = \{ \pi \in \{0,1\}^{\leq 2^n - f(n)} \mid U(\pi, n) \in \{0,1\}^{2^n} \text{ in } \leq 2^c \text{ space} \}. \quad (5.4)$$

For $n \in \mathbb{N}$ and $\pi \in B_n$, let

$$Z_{n,\pi} = \bigcup_{|z|=2^n-1} C_{zU(\pi, n)}. \quad (5.5)$$

(Thus $Z_{n,\pi}$ is the set of all languages $A$ such that $U(\pi, n)$ is the $2^n$-bit characteristic string of $A_{=n}$.) For $n \in \mathbb{N}$ and $w \in \{0,1\}^*$, let

$$\sigma(n, w) = \sum_{\pi \in B_n} \Pr(Z_{n,\pi} \mid C_w),$$

where the conditional probabilities $\Pr(Z_{n,\pi} \mid C_w) = \Pr[A \in Z_{n,\pi} \mid A \in C_w]$ are computed according to the random experiment in which a language $A \subseteq \{0,1\}^*$ is
chosen probabilistically, using an independent toss of a fair coin to decide membership of each string in $A$. Finally, define the function $d : \mathbb{N} \times \{0,1\}^* \rightarrow [0,\infty)$ as follows. (In all three clauses, $n \in \mathbb{N}$, $w \in \{0,1\}^*$, and $b \in \{0,1\}$.)

(i) If $0 < |w| < 2^n - 1$, then $d_n(w) = 2^{1-f(n)}$.

(ii) If $2^n - 1 < |w| < 2^{n+1} - 1$, then $d_n(wb) = d_n(w)\frac{\sigma(n,wb)}{\sigma(n,w)}$.

(iii) If $|w| \geq 2^{n+1} - 1$, then $d_n(wb) = d_n(w)$.

(The condition $\sigma(n, w) = 0$ can only occur if $d_n(w) = 0$, in which case we understand clause (ii) to mean that $d_n(wb) = 0$.)

It is clear from (5.5) that

$$\sigma(n, w) = \frac{\sigma(n, w0) + \sigma(n, w1)}{2}$$

for all $n \in \mathbb{N}$ and $w \in \{0,1\}^*$. It follows by a routine induction on the definition of $d$ that $d$ is a 1-DS. It is also routine to check that $d$ is pspace-computable. (The crucial point here is that we are only required to perform computations of the type (5.5) when $|w| \geq 2^n - 1$, so the $2^n$ space bound of (5.4) is polynomial in $|w|$.) Since $\sum_{n=0}^{\infty} 2^{-f(n)}$ is $p$-convergent, it is immediate from clause (i) that (5.2) holds. All that remains, then, is to verify (5.3).

For each language $A \subseteq \{0,1\}^*$, let

$$I_A = \{n \in \mathbb{N} \mid KS^{2^n}(A_{=n}) \leq 2^n - f(n)\}.$$ 

Fix a language $A$ for a moment and let $n \in I_A$. Then there exists $\pi_0 \in B_n$ such that $A \in Z_n,\pi_0$. Fix such a program $\pi_0$ and let $x, y \in \{0,1\}^*$ be the characteristic strings
of $A_{<n}$, $A_{\leq n}$, respectively. (Thus $|x| = 2^n - 1$, $|y| = 2^{n+1} - 1$, and $y = xU(\pi_0, n)$.) The definition of $d$ tells us that $d_n(y)$ is $d_n(x)$ times a telescoping product, i.e.,

$$d_n(y) = d_n(x) \prod_{i=0}^{2^{n-1}} \frac{\sigma(n,y[0..2^n+1])}{\sigma(n,y[0..2^{n+1}-1+1])}$$

$$= d_n(x) \frac{\sigma(n,y)}{\sigma(n,x)}$$

$$= 2^{1-f(n)} \frac{\sigma(n,y)}{\sigma(n,x)}. \quad (5.6)$$

Since $C_y \subseteq Z_{n,\pi_0}$, we have

$$\sigma(n, y) = \sum_{\pi \in B_n} \Pr(Z_{n,\pi} \mid C_y) \geq \Pr(Z_{n,\pi_0} \mid C_y) = 1. \quad (5.7)$$

For each $\pi \in B_n$, the events $C_\pi$ and $Z_{n,\pi}$ are independent, so

$$\sigma(n, x) = \sum_{\pi \in B_n} \Pr(Z_{n,\pi} \mid C_\pi)$$

$$= \sum_{\pi \in B_n} \Pr(Z_{n,\pi})$$

$$= |B_n|2^{-2^n} \quad (5.8)$$

< 2^{1-f(n)}.$$

By (5.6), (5.7), and (5.8), we have $d_n(y) > 1$. It follows that $A \in C_y \subseteq S[d_n]$. Since $n \in I_A$ is arbitrary here, we have shown that $A \in S[d_n]$ for all $A \subseteq \{0,1\}^*$ and $n \in I_A$. It follows that, for all $A \subseteq \{0,1\}^*$,

$$A \in X^c \Rightarrow |I_A| = \infty \Rightarrow A \in S[d_n] \text{ i.o.} \Rightarrow A \in \bigcap_{t=0}^{\infty} \bigcup_{n=t}^{\infty} S[d_n],$$

i.e., (5.3) holds. This completes the proof. \qed
Corollary 5.5. Let \( c \in \mathbb{N} \) and \( \varepsilon > 0 \). If

\[
X = \{ A \subseteq \{0,1\}^* \mid KS^{2^c}(A_{mn}) > 2^n - n^\varepsilon \text{ a.e.} \},
\]

then \( \mu_{\text{pspace}}(X) = \mu(X \mid \text{ESPACE}) = 1 \).

**Proof.** Routine calculus shows that the series \( \sum_{n=0}^{\infty} 2^{-n^n} \) is p-convergent. \( \square \)

Corollary 5.5 is a substantial improvement of Theorem 5.1(a). We exploit this improvement throughout this chapter.

5.1.2 Nonuniform Complexity Cores

In Chapter 4 we show that almost every language in \( E \) has “large” uniform complexity cores. Here we show that almost every language in \( \text{ESPACE} \) has “large” nonuniform complexity cores. Nonuniform complexity cores were first defined and investigated by Huynh [36] with respect to the complexity class \( \text{P/Poly} \). Here we present a modified version of the original definition.

Given a machine \( M \), an advice function \( h \), and an input \( x \in \{0,1\}^* \), we write \( M/h(x) = 1 \) if \( M \) accepts \( \langle x, h(|x|) \rangle \), \( M/h(x) = 0 \) if \( M \) rejects \( \langle x, h(|x|) \rangle \), and \( M(x) = \perp \) in any other case (i.e., if \( M \) fails to halt or \( M \) halts without deciding \( \langle x, h(|x|) \rangle \)). If \( M(x) \in \{0,1\} \), we write \( \text{space}_{M/h}(x) \) for the number of steps used in the computation of \( M((x, h(|x|))) \). If \( M(x) = \perp \), we define \( \text{space}_{M/h}(x) = \infty \). We partially order the set \( \{0,1,\perp\} \) by \( \perp < 0 \) and \( \perp < 1 \), with 0 and 1 incomparable. A machine/advice pair \( M/h \) is consistent with a language \( A \subseteq \{0,1\}^* \) if \( M/h(x) \leq [x \in A] \) for all \( x \in \{0,1\}^* \).
Definition (Huynh [36]). Let \( s : \mathbb{N} \to \mathbb{N} \) be a space bound and let \( A, K \subseteq \{0,1\}^* \). Then \( K \) is a \( \text{DSPACE}(s(n))/\text{Poly-complexity core} \) of \( A \) if, for every \( c \in \mathbb{N} \) the following holds. For every machine \( M \) and polynomially bounded advice function \( h \), if \( M/h \) is consistent with \( A \), then the fast set 
\[
F = \{ x \mid \text{space}_{M/h}(x) \leq c \cdot s(|x|) + c \}
\]
has the property that \( |F \cap K| \) is sparse.

Intuitively, very complex languages must have large nonuniform complexity cores. This intuition is supported by the following technical lemma.

Lemma 5.6. If \( s : \mathbb{N} \to \mathbb{N} \) is space constructible and \( p \) is a polynomial, then every language \( A \) with 
\[
KS^{s(n)}(A_{=n}) > 2^n - p(n) \text{ a.e.}
\]
has \( \{0,1\}^* \) as a \( \text{DSPACE}(s(n))/\text{Poly-complexity core} \).

Proof. We show the contrapositive. Assume that \( A \) does not have \( \{0,1\}^* \) as a \( \text{DSPACE}(s(n))/\text{Poly-complexity core} \). Under this assumption, there exist a machine \( M \), polynomially bounded advice function \( h \), and constant \( c \) such that \( M \) given \( h \) is consistent with \( A \) and the set 
\[
F = \{ x \mid \text{space}_{M/h}(x) \leq c \cdot s(|x|) + c \}
\]
is non-sparse. Using \( M \), \( c \), and a machine for \( s \), we construct a machine \( M' \) to output \( \chi_{A_{=n}} \) as in Figure 5.1.

Now consider the action of \( M' \) on input \((h(n),y,n)\), where \( y \) is the string \( A_{=n} \) with the bits corresponding to the elements of \( F_{=n} \) removed. On this input, the
Figure 5.1: An algorithm that computes $\chi_{A_{\infty}}$.

Machine $M'$ correctly outputs the bits of $A_{\infty}$ either (1) by deciding $[s_i \in A]$ directly or (2) by using the bits of $y$. Thus we have the following.

$$KS^{s(n)+c}(A_{\infty}) \leq |(h(n), y)|$$
$$\leq 2|h(n)| + 2 + |A_{\infty}| - |F_{\infty}|$$
$$\leq 2|h(n)| + 2 + 2^n - |F_{\infty}|$$

By universal simulation, there exists a constant $c_1 \in \mathbb{N}$ such that

$$KS^{c_1(s(n)+c)+c_1}(A_{\infty}) \leq KS^{s(n)+c}(A_{\infty}) + c_1$$
$$\leq 2^n - |F_{\infty}| + 2|h(n)| + 2 + c_1.$$ 

The above inequality combined with the fact that $F$ is non-sparse proves that

$$KS^{n-s(n)}(A_{\infty}) \leq KS^{c_1(s(n)+c)+c_1}(A_{\infty}) < 2^n - p(n) \text{ i.o.}$$

for every polynomial $p$. \qed
Since almost every language in ESPACE has high space-bounded Kolmogorov complexity almost everywhere, Lemma 5.6 allows us to conclude that almost every language in ESPACE has maximal nonuniform complexity cores.

**Corollary 5.7.** Fix $c \in \mathbb{N}$. Then, almost every language in ESPACE has $\{0,1\}^*$ as a $\text{DSPACE}(2^c)/\text{Poly}$-complexity core.

**Proof.** By Corollary 5.5, the set

$$X = \{ A \subseteq \{0,1\}^* | KS^{2^{(c+1)n}}(A_m) > 2^n - \sqrt{n} \text{ a.e.} \}$$

has pspace-measure 1. By Lemma 5.6, each element of $X$ has $\{0,1\}^*$ as a $\text{DSPACE}(2^c)/\text{Poly}$-complexity core. It follows that almost every language in ESPACE has $\{0,1\}^*$ as a $\text{DSPACE}(2^c)/\text{Poly}$-complexity core. \hfill \Box

### 5.1.3 Incompressibility by Nonuniform Reductions

In Chapter 4 we also show that almost every language in $E$ is incompressible by $\leq_m^p$-reductions. Here we extend this result to show that almost every language in ESPACE is $n^{\log n}$-incompressible by $\leq_m^{p/poly}$-reductions. First some notation is necessary.

Recall from Chapter 4 that the *collision set* of a function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is

$$C_f = \{ x \in \{0,1\}^* | (\exists y < x)f(y) = f(x) \}.$$  

A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is *one-to-one almost everywhere* (or, briefly, *one-to-one a.e.*) if its collision set $C_f$ is finite. The following definitions generalize the notion of incompressibility presented in Chapter 4.
**Definition.** Let $F$ be a function class. Then a language $A \subseteq \{0, 1\}^*$ is incompressible by $\leq_F$-reductions if every $\leq_m$-reduction of $A$ is one-to-one a.e.

**Definition.** Let $g : \mathbb{N} \rightarrow \mathbb{N}$ and let $F$ be a function class. Then a language $A$ is $g(n)$-incompressible by $\leq_F$-reductions if every $\leq_m$-reduction $f$ of $A$ satisfies $|\langle C_f \rangle|_n \leq g(n)$ for almost all $n$.

The above generalization of incompressibility is necessary here. Notice that no language is incompressible by $\leq_m^{p/poly}$-reductions. Moreover, if $p$ is a polynomial then no language is $p(n)$-incompressible by $\leq_m^{p/poly}$-reductions. However, if $g$ is efficiently computable and exceeds every polynomial almost everywhere then almost every language in ESPACE is $g(n)$-incompressible by $\leq_m^{p/poly}$-reductions. (For notational convenience we say that such $g$ are superpolynomial. More precisely, a function $g : \mathbb{N} \rightarrow \mathbb{N}$ is superpolynomial if for every polynomial $p$, $g(n) \geq p(n)$ a.e.)

**Theorem 5.8.** Fix $c \in \mathbb{Z}^+$. Let $g : \mathbb{N} \rightarrow \mathbb{N}$ be superpolynomial and in $\text{DSPACE}(2^n)$. If we let

$$X = \{A \subseteq \{0, 1\}^* \mid A \text{ is } g(n)\text{-incompressible by } \leq_m^{\text{DSPACE}(2^n)/poly}\text{-reductions}\},$$

then $\mu_{\text{pspace}}(X) = 1$. It follows that $\mu(X \mid \text{ESPACE}) = 0$, i.e., almost every language in ESPACE is $g(n)$-incompressible by $\leq_m^{\text{DSPACE}(2^n)/poly}$-reductions.

**Proof.** We follow the format of the proof for Theorem 5.4. Assume the hypothesis. By Lemma 2.2, it suffices to exhibit a pspace-computable 1-DS $d$ such that

$$\sum_{n=0}^{\infty} d_n(\lambda) \text{ is } p\text{-convergent} \quad (5.9)$$
Some notation will be helpful for the remainder of the proof. For \( n \in \mathbb{N} \) and \( g : \mathbb{N} \to \mathbb{N} \), let

\[
M/Adv(g)(n) = \left\{ f : \exists n_0 \leq n \text{ and } \exists h_0, \ldots, h_n \in \{0,1\}^{g(n)} \text{ such that} \forall x \in \{0,1\}^{\leq n}, M_{h_0}(x, h_{[x]}) = f(x) \text{ in } \leq 2^m \text{ space.} \right\}.
\]

Informally, the class \( M/Adv(g)(n) \) is the set of functions that are computed correctly over \( \{0,1\}^{\leq n} \) by one of the first \( n \) machines with advice bounded in length by \( g(n) \).

For \( n \in \mathbb{N} \), let

\[
B_n = \{ f \in M/Adv\left(\frac{g(n)}{4n^2} - 3\right)(n) \mid |(C_f)_{=n}| \geq \frac{g(n)}{2n} \}. \tag{5.11}
\]

For \( n \in \mathbb{N} \) and \( f \in B_n \), let

\[
Y_{n,f} = \{ x \in \{0,1\}^{2^{n+1}-1} \mid \forall i, j \leq 2^{n+1} - 1, f(s_i) = f(s_j) \Rightarrow x[i] = x[j] \}
\]

and

\[
Z_{n,f} = \bigcup_{x \in Y_{n,f}} C_x.
\]

(Thus \( Z_{n,f} \) is the set of all languages \( A \) such that no counterexample to the statement "\( f \) is a many-one reduction of \( A \)" exists among the strings in \( \{0,1\}^{\leq n} \).) For \( n \in \mathbb{N} \) and \( w \in \{0,1\}^* \), let

\[
\sigma(n, w) = \sum_{f \in B_n} \Pr(Z_{n,f} \mid C_w), \tag{5.12}
\]

where the conditional probabilities \( \Pr(Z_{n,f} \mid C_w) = \Pr_{A}[A \in Z_{n,f} \mid A \in C_w] \) are computed according to the random experiment in which a language \( A \subseteq \{0,1\}^* \) is
chosen probabilistically, using an independent toss of a fair coin to decide membership of each string in $A$. Finally, define the function $d : \mathbb{N} \times \{0,1\}^* \to [0,\infty)$ as follows.

(In all three clauses, $n \in \mathbb{N}$, $w \in \{0,1\}^*$, and $b \in \{0,1\}$.)

(i) If $0 < |w| < 2^n - 1$, then $d_n(w) = 2^{-n}$.

(ii) If $2^n - 1 < |w| < 2^{n+1} - 1$, then $d_n(wb) = d_n(w)\frac{\sigma(n,wb)}{\sigma(n,w)}$.

(iii) If $|w| \geq 2^{n+1} - 1$, then $d_n(wb) = d_n(w)$.

(Note that the condition $\sigma(n,w) = 0$ can only occur if $d_n(w) = 0$, in which case we understand clause (ii) to mean that $d_n(wb) = 0$.)

It is clear from (5.12) that

$$\sigma(n,w) = \frac{\sigma(n,w0) + \sigma(n,w1)}{2}$$

for all $n \in \mathbb{N}$ and $w \in \{0,1\}^*$. It follows by a routine induction on the definition of $d$ that $d$ is a 1-DS. It is also routine to check that $d$ is pspace-computable. Furthermore, since the sum $\sum_{n=0}^{\infty} 2^{-n}$ is $p$-convergent, it is immediate from clause (i) that (5.9) holds. All that remains is to verify (5.10).

Let $A \in \mathcal{X}^c$ and fix $f \in \text{DSPACEF}(2^m)/\text{Poly}$ such that $f$ is a many-one reduction of $A$ with $|(C_f)_{\leq n}| > g(n)$ i.o. Define the set

$$I_{A,f} = \{n \in \mathbb{N} \mid f \in M/Adv\left(\frac{g(n)}{4n^2} - 3\right)(n) \text{ and } |(C_f)_{=n}| \geq \frac{g(n)}{2n}\}.$$

Since $g$ is superpolynomial and $f$ has $|(C_f)_{\leq n}| > g(n)$ i.o., it follows that $I_{A,f}$ is infinite. Let $n \in I_{A,f}$ and let $x, y \in \{0,1\}^*$ be the characteristic strings of $A_{\leq n}$, $A_{\leq n}$, respectively. The definition of $d$ tells us that $d_n(y)$ is $d_n(x)$ times a telescoping
Let \( d_n(y) \) be a product, i.e.,

\[
d_n(y) = d_n(x) \frac{\sigma(n,y)}{\sigma(n,x)}
\]

since \( C_y \subseteq Z_{n,f} \), we have

\[
\sigma(n,y) = \sum_{f \in B_n} \Pr(Z_{n,f} \mid C_y) \geq \Pr(Z_{n,f} \mid C_y) = 1.
\]

Now a simple counting argument shows that there are at most \( 2^{\frac{g(n)}{4n}} \) functions in \( B_n \) that are distinct functions over \( \{0,1\}^n \). Furthermore, for each \( f \in B_n \), there are at most \( 2^{2n-\frac{(C_f)n}{2}} \) possible \( 2^n \) bit extensions of \( x \) satisfying \( f \). (That is, there are at most \( 2^{2n-\frac{(C_f)n}{2}} \) strings \( z \) such that \( xz \in Y_{n,f} \).) Thus we have \( \Pr(Z_{n,f} \mid C_x) \leq 2^{-\frac{(C_f)n}{2}} = 2^{-\frac{g(n)}{4n}} \), so

\[
\sigma(n,x) = \sum_{f \in B_n} \Pr(Z_{n,f} \mid C_x) \leq \sum_{f \in B_n} 2^{-\frac{g(n)}{4n}} = |B_n|2^{-\frac{g(n)}{4n}} < 2^{-n}.
\]

By (5.13), (5.14), and (5.15), we have \( d_n(y) > 1 \). It follows that \( A \in C_y \subseteq S[d_n] \). Since \( n \in I_{A,f} \) is arbitrary here, we have shown that \( A \in S[d_n] \) for all \( A \in X^c \text{ and } \sum_{n \in I_{A,f}} \). It follows that, for all \( A \subseteq \{0,1\}^* \),

\[
A \in X^c \Rightarrow \exists f \text{ such that } |I_{A,f}| = \infty
\]

\[
\Rightarrow A \in S[d_n] \text{ i.o.}
\]

\[
\Rightarrow A \in \bigcap_{t=0}^{\infty} \bigcup_{n=1}^{\infty} S[d_n],
\]
i.e., (5.10) holds. This completes the proof.

**Corollary 5.9.** Almost every language in ESPACE is $n^{\log n}$-incompressible by \(\leq^p_{m\text{-}\text{poly}}\) reductions.

Corollary 5.9 guarantees the existence of an $n^{\log n}$-incompressible language \(A\) in ESPACE but does not allow us to specify a constant \(c\) such that \(A \in \text{DSPACE}(2^m)\). However, it is easy to show that such an \(A\) exists in DSPACE($2^{2n}$). This fact is useful in section 5.3.

**Claim 5.10.** There exists a language \(A\) that is $n^{\log n}$-incompressible by $\leq^p_{m\text{-}\text{poly}}$ reductions and contained in DSPACE($2^{2n}$).

### 5.2 Nonuniform Complexity of Weakly Hard Problems: Lower Bounds

In the previous section we saw that almost every language in ESPACE is maximally complex with respect to two measures of nonuniform complexity. In this section we show that every weakly \(\leq^p_{m\text{-}\text{poly}}\)-hard language for ESPACE obeys tight lower bounds on its complexity in terms of these complexity measures. (Recall that a language \(H\) is weakly \(\leq^p_{m\text{-}\text{poly}}\)-hard for ESPACE if the set

\[
P/\text{Poly}_m(H) = \{A \subseteq \{0, 1\}^* \mid A \leq^p_{m\text{-}\text{poly}} H\}
\]

does not have measure 0 in ESPACE.) Our first result shows that every weakly \(\leq^p_{m\text{-}\text{poly}}\)-hard language for ESPACE has a dense DSPACE($2^n$)-complexity core. These result extends work of Huynh [36].
The proof of Theorem 5.11 uses the following special notation. The nonreduced image of a language $S \subseteq \{0, 1\}^*$ under a function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is

$$f^2(S) = \{f(x) \mid x \in S \text{ and } |f(x)| \geq |x|\}.$$

Note that

$$f^2(f^{-1}(S)) = S \cap f^2(\{0, 1\}^*)$$

for all $f$ and $S$.

**Theorem 5.11.** For every weakly $\leq^p_m$-hard language $H$ for ESPACE, there is a real $\epsilon > 0$ such that $H$ has a dense $\text{DSPACE}(2^n)/\text{Poly}$ complexity core.

**Proof.** Let $H$ be weakly $\leq^p_m$-hard for ESPACE, let

$$X = \{A \subseteq \{0, 1\}^* \mid A \text{ is } n^{\log n}\text{-incompressible by } \leq_m^{\text{DSPACE}(2^n)}/\text{poly-reductions}\},$$

and let

$$Y = \{A \subseteq \{0, 1\}^* \mid A \text{ has } \{0, 1\}^* \text{ as a } \text{DSPACE}(2^n)/\text{Poly} \text{ complexity core}\}.$$

Notice that by Corollary 5.7 and Theorem 5.8 the set $X \cap Y$ has measure 1 in ESPACE. Moreover, since $P/\text{Poly}_m(H)$ does not have measure 0 in ESPACE, $X \cap Y \cap P/\text{Poly}_m(H)$ is not empty. Fix $A \in X \cap Y \cap P/\text{Poly}_m(H)$, let $f$ be a $\leq^p_m$-reduction of $A$ to $H$, let $q$ be a strictly increasing polynomial bound on the length of strings produced by $f$ (i.e., $q(n) \geq \max\{|f(x)| \mid x \in \{0, 1\}^{\leq n}\}$), and let $\epsilon = \frac{1}{3 \cdot \deg(q)}$.

We show that

$$K = f^2(\{0, 1\}^*)$$

is a dense $\text{DSPACE}(2^n)/\text{Poly}$ complexity core of $H$. 
By our choice of $c$, $q([n^{2c}]) < n$ for all sufficiently large $n$. Let $W = \{x | |f(x)| < |x|\}$. Then, for all sufficiently large $n \in \mathbb{N}$, writing $m = \lfloor n^{2c} \rfloor$, we have

$$f(\{0,1\}^{\leq m}) - \{0,1\}^{<m} \subseteq f(\{0,1\}^{\leq m}) - f(W^{\leq m})$$

$$\subseteq f^{\geq}(\{0,1\}^{\leq m})$$

$$\subseteq K_{\leq s(m)}$$

$$\subseteq K_{\leq n},$$

whence

$$|K_{\leq n}| \geq |f(\{0,1\}^{\leq m})| - |\{0,1\}^{<m}|$$

$$\geq |\{0,1\}^{\leq m}| - |(C_f)_{\leq m}| - |\{0,1\}^{<m}|$$

$$= 2^m - |(C_f)_{\leq m}|$$

$$\geq 2^{|n^{2c}|} - |(C_f)_{\leq n}|.$$ 

Since $|(C_f)_{\leq n}| < n^{\log n}$ a.e., it follows that $|K_{\leq n}| > 2^{n^c}$ for all sufficiently large $n$. Thus $K$ is dense.

To see that $K$ is a DSPACE$(2^{n^c})$/Poly complexity core of $H$, let $c \in \mathbb{N}$, let $M$ be a machine and $h$ be a polynomial advice function such that $M/h$ is consistent with $H$, and define the fast set

$$F = \{x | \text{space}_{M/h}(x) \leq c \cdot 2^{|x|} + c\}.$$ 

Let $\hat{M}/\hat{h}$ be a machine/polynomial advice pair (constructed in the obvious way) such that

$$\hat{M}/\hat{h}(x) = M/h(f(x))$$
for all \( x \in \{0,1\}^\ast \). Since \( f \) reduces \( A \) to \( H \) and \( M/h \) is consistent with \( H \), \( \hat{M}/\hat{h} \) is consistent with \( A \). Since \( A \) has \( \{0,1\}^\ast \) as a DSPACE(2\(^n\))/Poly complexity core, the fast set

\[
\hat{F} = \{ x \mid \text{space}_{\hat{M}/\hat{h}}(x) \leq c \cdot 2^n + c \}
\]

is sparse. By our choice of \( \epsilon \), \( y \in F \cap f(\{0,1\}^\ast) \) implies \( y \in f(\hat{F}) \) for all but finitely many \( y \). Since \( \hat{F} \) is sparse, we have

\[
|F \cap K|_{\leq n} = |(F \cap \hat{F}(\{0,1\}^\leq n))_{\leq n}| \\
\leq |(\hat{F} \cap \{0,1\}^\leq n)_{\leq n}| + c \\
\leq |\hat{F} \cap \{0,1\}^\leq n| + c \\
\leq p(n) + c.
\]

Hence \( F \cap K \) is sparse. Thus \( K \) is DSPACE(2\(^n\))/Poly complexity core of \( H \). \( \square \)

**Corollary 5.12** (Huynh [36]). Every \( \leq_P \)-hard language for ESPACE has a dense \( P/Poly \)-complexity core. \( \square \)

Our next theorem provides a lower bound on the Kolmogorov complexity of weakly \( \leq_P \)-hard languages for ESPACE. This result extend another result of Huynh [34].

**Theorem 5.13.** For every weakly \( \leq_P \)-hard language \( H \) for ESPACE, there exists an \( \epsilon > 0 \) such that

\[
KS^{2^n}(H_{\leq n}) > 2^n \text{ a.e.}
\]

**Proof.** Let \( H \) be weakly \( \leq_P \)-hard for ESPACE and let

\[
X = \{ A \subseteq \{0,1\}^\ast \mid KS^{2^n}(A_m) > 2^n - \sqrt{n} \text{ a.e.} \}.
\]
Since $\mathbb{P}/\text{Poly}_T(H)$ does not have measure 0 in $\text{ESPACE}$ and $X$ has measure 1 in $\text{ESPACE}$ (by Corollary 5.5), the set $\mathbb{P}/\text{Poly}_T(H) \cap X \cap \text{ESPACE}$ is not empty. Fix $A \in \mathbb{P}/\text{Poly}_m(H) \cap X$ and let $M_A/h$ be an oracle machine/polynomial advice pair that decides $A$ in polynomial time using $H$ as an oracle. Moreover, fix $k \in \mathbb{N}$ such that the computation $M_A(x, h(|x|))$ queries the oracle on strings of length at most $|x|^k$ for almost every $x$ and let $\epsilon = \frac{1}{3^k}$. We will essentially show that

$$KS^{2n^k}(H_{\leq n}) \geq KS^{2^n|n^{2k}|}(A_{\leq |n^{2k}|}) > 2^{n^k} \text{ a.e.}$$

Let $\hat{M}_A$ be a machine that efficiently implements the algorithm in Figure 5.2, let $n$ be sufficiently large, and let $\pi$ be a minimal $2^n$-space bounded program for $X_{H_{\leq n}}$. Then the machine $\hat{M}_A$ on input $(\langle h(0), h(1), \ldots, h(m) \rangle \pi, m)$, where $m = \lceil n^{2k} \rceil$, outputs $X_{A_{\leq m}}$ using less than $c_0 \cdot 2^{n^k}$ space. From our previous observation and universal simulation, we have that

$$KS^{2^{2m}}(A_{\leq m}) \leq KS^{2^{2n^k}}(A_{\leq m}) + c_1$$

$$\leq |\langle h(0), h(1), \ldots, h(m) \rangle \pi| + c_1$$

$$\leq KS^{2n^k}(H_{\leq n}) + |\langle h(0), \ldots, h(m) \rangle| + c_1 \text{ a.e.}$$

Since $KS^{2^{2m}}(A_{\leq m}) > 2^m - \sqrt{m}$ a.e., and the length of $\langle h(0), \ldots, h(m) \rangle$ is bounded by some polynomial $q$, it follows that

$$KS^{2n^k}(H_{\leq n}) \geq KS^{2^{2m}}(A_{\leq m}) - q(m) - c_1$$

$$\geq 2^{n^{2k}} - \sqrt{|n^{2k}|} - q(|n^{2k}|) - c_1$$

$$> 2^{n^k} \text{ a.e.}$$

This completes the proof.
\[ \tilde{M}_A((h_0, \ldots, h_n)\pi, n) : \]
\begin{algorithm}
    \textbf{begin}
    \textbf{for} each \( s_i \in \{0, 1\}^n \) \textbf{do}
    \begin{algorithm}
    \textbf{begin}
    (1) Simulate \( M_A(s_i, h_{|s_i|}) \) as usual, but
    when \( M_A \) queries the oracle on \( s_j \) perform (1.1)
    (1.1) Simulate \( U(\pi, n) \) and dispose of the output
    until the \( j^{th} \) bit is written.
    \textbf{if} the \( j^{th} \) bit is a '0' \textbf{then} continue (1) as if
    the oracle said "No".
    \textbf{if} the \( j^{th} \) bit is a '1' \textbf{then} continue (1) as if
    the oracle said "Yes".
    (2) When \( M_A(s_i, h_{|s_i|}) \) halts and accepts, rejects,
    write a '1', '0', respectively to the output tape.
    \textbf{end for}
    \textbf{end}
    \end{algorithm}
    \textbf{end for}
\end{algorithm}

Figure 5.2: The algorithm for \( \tilde{M}_A \) in the proof of Theorem 5.13.

Corollary 5.14 (Huynh[34]). For every \( \leq^p_T \)-hard language \( H \) for ESPACE, there exists an \( \epsilon > 0 \) such that
\[ KS^{2^n}(H_\leq_n) > 2^n \text{ a.e.} \]

The remaining results of this section show that Theorems 5.11 and 5.13 cannot be significantly improved.

Theorem 5.15. For every \( \epsilon > 0 \), there is a \( \leq^p_m \)-complete language \( C \) for ESPACE such that each DSPACE(\( 2^n \))/Poly complexity core of \( C \) is sparse.
Proof. Let $C$ be $\leq^p_m$-complete for ESPACE and decidable in DSPACE($2^{n^e}$). Since $C$ can be decided in $2^{n^e}$ space every DSPACE($2^{n^e}$/Poly complexity core of $C$ must be sparse.

\begin{theorem}
For every $\epsilon > 0$, there is a $\leq^p_m$-complete language $C$ for ESPACE such that
\[ KS^{2^{n^e}}(C_{=n}) \leq c \text{ a.e.} \]
and
\[ KS^{2^{n^e}}(C_{\leq n}) \leq c \text{ a.e.} \]
for some constant $c \in \mathbb{N}$.
\end{theorem}

\begin{proof}
Let $C$ be $\leq^p_m$-complete for ESPACE and decidable in DSPACE($2^{n^{e/2}}$). Since $C$ can be decided in $2^{n^{e/2}}$ space there are constant programs $\pi_0, \pi_1$ such that

1. $U(\pi_0, n) = \chi_{C_{=n}}$ in less than $2^{n^e}$ space, and

2. $U(\pi_1, n) = \chi_{C_{\leq n}}$ in less than $2^{n^e}$ space.
\end{proof}

5.3 Nonuniform Complexity of Hard Problems: Upper Bounds

In this section we establish tight upper bounds on the Kolmogorov complexity and size of nonuniform complexity cores for $\leq_{m}^{p/poly}$-hard problems for ESPACE. We first prove the following technical lemma that says if $A \in$ ESPACE is incompressible by $\leq_{m}^{p/poly}$-reductions, then any language that $A$ is reducible to has a dense "easily decidable" subset.
Lemma 5.17. If $A$ is decidable in $\text{DSPACE}(2^{cn})$, is $n^{\log n}$-incompressible by $\leq^p_{\text{poly}}$-reductions, and is $\leq^p_{\text{poly}}$-reducible to $H$, then there exist $B, D \in \text{DSPACE}(2^{cn})/\text{Poly}$ such that $D$ is dense and $B = H \cap D$.

Proof. Assume the hypothesis and let $f$ be a $\leq^p_{\text{poly}}$-reduction of $A$ to $H$. Let $B = f^2(A)$ and $D = f^2(\{0,1\}^*)$. (Recall from section 5 that $f^2(S) = \{f(x) \mid x \in S$ and $|f(x)| \geq |x|\}$.) Since $A \in \text{DSPACE}(2^{cn})$ and $f \in \text{P/Pol}y$, it is clear that $B, D \in \text{DSPACE}(2^{cn})/\text{Poly}$. Furthermore, it is clear that $D$ is dense from the argument given in the proof of Theorem 5.11. Finally, note that $B = f^2(A) = f^2(f^{-1}(H)) = H \cap f^2(\{0,1\}^*) = H \cap D$. \qed

Corollary 5.18. If $A$ is decidable in $\text{DSPACE}(2^{cn})$, $n^{\log n}$-incompressible by $\leq^p_{\text{poly}}$-reductions, and is $\leq^p_{\text{poly}}$-reducible to $H$, then every $\text{DSPACE}(2^{cn})/\text{Poly}$-complexity core $K$ of $H$ has a dense complement. \qed

Since there exist languages in $\text{DSPACE}(2^{2n})$ that are $n^{\log n}$-incompressible by $\leq^p_{\text{poly}}$-reductions (Claim 5.10), we get the following theorem and its corollary concerning the complexity of $\leq^p_{\text{poly}}$-hard languages for $\text{ESPACE}$.

Theorem 5.19. For every $\leq^p_{\text{poly}}$-hard language $H$ for $\text{ESPACE}$, there exist $B, D \in \text{DSPACE}(2^{2n})/\text{Poly}$ such that $D$ is dense and $B = H \cap D$. \qed

It follows immediately that every $\text{DSPACE}(2^{2n})/\text{Poly}$-complexity core of every $\leq^p_{\text{poly}}$-hard language for $\text{ESPACE}$ has a dense complement.

Corollary 5.20. Every $\text{DSPACE}(2^{2n})/\text{Poly}$-complexity core $K$ of every $\leq^p_{\text{poly}}$-hard language for $\text{ESPACE}$ has a dense complement. \qed
\begin{figure}
\begin{verbatim}
M((u,v)y,n);
begin
  z = \bot 2^n;
  for i = 0 to 2^n - 1 do
  begin
    if MD(w_i, u) accepts then
      simulate MB(w_i, v);
    if this simulation accepts or rejects
      then set z[i] = 1 or z[i] = 0, respectively
    else
      (z[i], y) = (head(y), tail(y));
  end;
output z;
end.
\end{verbatim}
\caption{The machine M}
\end{figure}

Theorem 5.19 says that every \textit{\leq}_{m}^{p/poly}-hard language is \textit{DSPACE}(2^{2n})/\textit{Poly} decidable on a dense, \textit{DSPACE}(2^{2n})/\textit{Poly} decidable set of inputs. We exploit Theorem 5.19 to show that every \textit{\leq}_{m}^{p/poly}-hard language for ESPACE has unusually low space-bounded Kolmogorov complexity infinitely often.

\textbf{Theorem 5.21.} For every \textit{\leq}_{m}^{p/poly}-hard language \( H \) for ESPACE, there exists an \( \epsilon > 0 \) such that

\[ KS^{2^{2n}}(H_{\epsilon n}) < 2^n - 2^{n^\epsilon} \text{ i.o.} \]

\textbf{Proof.} Let \( H \) be \textit{\leq}_{m}^{p/poly}-hard for ESPACE and fix \( B, D \) as in Theorem 5.19. Let the machines \( M_B, M_D \) and the advice functions \( h_B, h_D \) witness that \( B, D \in \textit{DSPACE}(2^{2n})/\textit{Poly} \) and fix \( \epsilon > 0 \) such that \( |D_{\epsilon n}| > 2^{n^\epsilon} \) i.o.

Let \( M \) be a machine that efficiently implements the algorithm in Figure 5.3 and
let $y_n$ be the string $\chi_{H_n}$ with the bits corresponding to $D_n$ removed. Then, the machine $M$ on input $((h_D(n), h_B(n))y_n, n)$ outputs the string $\chi_{H_n}$ and uses less than $O(2^n)$ space. It follows that

$$KS^{2^n}(H_n) \leq KS^{2^n+c}(H_n) + c$$

$$\leq |(h_D(n), h_B(n))y_n| + c$$

$$\leq 2^n - |D_n| + |(h_D(n), h_B(n))| + c.$$ 

Because both $h_B$ and $h_D$ are bounded in length by a polynomial, there is a polynomial $p$ such that $|(h_D(n), h_B(n))| \leq p(n)$. Thus, we have

$$KS^{2^n}(H_n) < 2^n - |D_n| + p(n) + c \leq 2^n - 2^n' \text{ i.o.}$$

Our next theorem illustrates that Corollary 5.20 can not be significantly improved.

**Theorem 5.22.** For every $\epsilon > 0$, there exists a $\leq_{m^D}$-complete language $C$ for ESPACE with a $\text{DSPACE}(2^n)/\text{Poly}$ complexity core $K$ with density

$$|K_{\leq n}| \geq 2^{n+1} - 2^n' \text{ a.e.}$$

**Proof.** Fix $\epsilon > 0$ and $k \in \mathbb{N}$ such that $\epsilon > \frac{1}{k} > 0$, let $A$ be $\leq_{m^D}$-complete for ESPACE, and fix $D \in \text{ESPACE}$ such that $D$ has $\{0,1\}^*$ as a $\text{DSPACE}(2^n)/\text{Poly}$ complexity core. Let $B = \{0^{\leq k}11x \mid x \in A\}$, let $K = \{0,1\}^* - \{0^{\leq k}11x \mid x \in \{0,1\}^*\}$, and define $C = (D \cap K) \cup B$. Since $B$ is $\leq_{m^D}$-complete for ESPACE, $K$ is decidable in
polynomial time, and $K \cap B$ is empty, it is clear that $C$ is $\leq^p_m$-complete for ESPACE. Moreover, notice that

$$|K_{\leq n}| = |\{0,1\}^n - \{0^{\leq|z|}11z | x \in \{0,1\}^*\}_{\leq n}|$$

$$= 2^{n+1} - 1 - |\{0^{\leq|z|}11z | x|^{k} + 2 + |x| \leq n\}$$

$$\geq 2^{n+1} - 2^{n^t + 1}$$

$$\geq 2^{n+1} - 2^{n^t} \text{ a.e.}$$

Thus it suffices to show that $K$ is DSPACE$(2^{2n})$/Poly complexity core of $C$.

Let $M/h$ be a machine/polynomial advice pair that is consistent with $C$, let $c$ be a constant, and define the fast set

$$F = \{x \in \{0,1\}^* | space_{M/h}(x) \leq c \cdot 2^{2|x|} + c\}.$$ 

Let $M$ be a machine (designed in the obvious way) such that

$$M_h(x) = \begin{cases} \bot & \text{if } x \in K^c \\ M/h(x) & \text{otherwise,} \end{cases}$$

and define the fast set

$$\hat{F} = \{x \in \{0,1\}^* | space_{\hat{M}/h}(x) \leq (c + 1) \cdot 2^{2|x|} + c\}.$$ 

Since membership in $K^c$ is decidable in polynomial time, it is clear that $F \Delta \hat{F} \cap K$ is finite. Furthermore, since $M$ is consistent with $D$, $\hat{F}$ is sparse. Since

$$F \cap K = (F \cap \hat{F}^c \cap K) \cup (F \cap \hat{F} \cap K)$$

$$\subseteq (F \Delta \hat{F} \cap K) \cup (F \cap \hat{F} \cap K)$$

$$\subseteq (F \Delta \hat{F} \cap K) \cup \hat{F},$$
it follows that $F \cap K$ is sparse. Thus $K$ is a DSPACE($2^{2n}$)/Poly complexity core of $C$.

\[\square\]

As the following theorem illustrates, Theorem 5.21 can not be significantly improved either.

**Theorem 5.23.** For every $\epsilon > 0$, there exists a $\leq_P$-complete language $C$ for ESPACE such that

\[ KS^{2^{2n}}(C^{=n}) > 2^n - 2^{n^\epsilon} \text{ a.e.} \]

**Proof.** Fix $\epsilon > 0$ and $k \in \mathbb{N}$ such that $\epsilon > \frac{1}{k} > 0$. Let $A$ be $\leq_m$-complete for ESPACE, let $B = \{0|\#^k 11x \mid x \in A\}$, and let $K = \{0|\#^k 11x \mid x \in \{0,1\}^*\}$. Notice that $B$ is $\leq_P$-complete for ESPACE and that $K$ is decidable in polynomial time. Now, construct $C$ in stages as in Figure 5.4. Since $C \cap K = B$, it is clear that $C$ is $\leq_P$-complete for ESPACE. It suffices to show that

\[ KS^{2^{2n}}(C^{=n}) > 2^n - 2^{n^\epsilon} \text{ a.e.} \]

Notice that there are $|P(K^{=n})|$ subsets $C^n$ of $\{0,1\}^n$ that satisfy $C^{=n} \cap K^{=n} = B^{=n}$. Since $|P(K^{=n})| > 2^{2^n - 2^{n^\frac{1}{k}}}$, for almost every $n$, there is a some set $C^n$ such that $C^n \cap K^{=n} = B^{=n}$ and no string $x$ in $\{0,1\}^{2^n - 2^{n^\frac{1}{k}}} - 2$ produces $\chi_{C^n}$ in $\leq 2^{2n}$ space. Hence, we have

\[ KS^{2^{2n}}(C^{=n}) = KS^{2^{2n}}(C^n) > 2^n - 2^{n^\frac{1}{k}} - 2 \text{ a.e.} \]

\[ > 2^n - 2^{n^\epsilon} \text{ a.e.} \]

\[\square\]
Figure 5.4: The construction of an $\leq_{m}^{P}$-complete language with high KS a.e.

5.4 Distribution of Hardness and Measure of Degrees

We now present the main result of this chapter. As we see from Corollaries 5.5 and 5.7, almost every language in ESPACE has the set $\{0, 1\}^{*}$ as a DSPACE($2^{cn}$)/Poly complexity core and has very high space-bounded Kolmogorov complexity almost everywhere. In Corollary 5.20 and Theorem 5.21, we show that the $\leq_{m}^{p/poly}$-hard languages do not satisfy these conditions. We conclude that the $\leq_{m}^{p/poly}$-hard languages are unusually simple.

**Theorem 5.24.** Let $C_{\text{ESPACE}}$, $C_{\text{ESPACE}}$ be the sets of languages that are $\leq_{m}^{p/poly}$-hard, $\leq_{m}^{p/poly}$-complete, respectively, for ESPACE. Then $C_{\text{ESPACE}}$ has pspace-measure 0, and hence $C_{\text{ESPACE}}$ is a measure 0 subset of ESPACE.
Proof. By Theorem 5.21, the set $\mathcal{H}_{\text{ESPACE}}$ is a subset of

$$X = \{ L \mid KS^{2^n}(L_{\equiv n}) \leq 2^n - \sqrt{n} \text{ i.o.}\}.$$ 

By Corollary 5.5, we have $\mu_{\text{pspace}}(X^c) = \mu(X^c \mid \text{ESPACE}) = 1$. It follows that $\mathcal{H}_{\text{ESPACE}}$ has pspace-measure 0, and hence $C_{\text{ESPACE}}$ is a measure 0 subset of ESPACE.

A closer examination of the results leading up to and including Theorem 5.24 reveals a surprising fact, namely, that there is a fixed language $A$ with the property that the set of languages that $A$ is $\leq_m^{p/poly}$-reducible to has measure 0 in ESPACE. This fact is a special case of the following more general phenomenon. For any fixed language $A \in \text{ESPACE}$, either the set of languages that $A$ is $\leq_m^{p/poly}$-reducible to or the set of languages that are $\leq_m^{p/poly}$-reducible to $A$ has measure 0 in ESPACE. We refer to this phenomenon as the "small span phenomenon."

The remaining results in this chapter precisely quantify the small span phenomenon. Recall from Chapter 2 that we write

$$P/\text{Poly}_m(L) = \{ A \subseteq \{0,1\}^* \mid A \leq_m^{p/poly} L \}$$

and

$$P/\text{Poly}_m^{-1}(L) = \{ A \subseteq \{0,1\}^* \mid L \leq_m^{p/poly} A \}$$

for the lower and upper $P/\text{Poly}$ many-one reducibility spans for a language $L$. Moreover, we write

$$\text{deg}_m^{p/poly}(L) = P/\text{Poly}_m(L) \cap P/\text{Poly}_m^{-1}(L)$$

for the $P/\text{Poly}$ many-one degree of $L$. 

Our first result generalizes the results leading up to and including Theorem 5.24. Recall that we use the existence of a language $A \in \text{DSPACE}(2^{2^n})$ that is $n \log n$-incompressible by $\leq^p_{\text{poly}}$-reductions to show that every $\text{DSPACE}(2^{2^n})/\text{Poly}$ complexity core of every $\leq^p_{\text{poly}}$-hard language has a dense complement. This result in combination with Corollary 5.7 immediately implies Theorem 5.24. In essence, we prove that the set $\text{P/Poly}^{-1}(A)$ has pspace-measure 0. This fact generalizes to all languages in $\text{ESPACE}$ that are $n \log n$-incompressible by $\leq^p_{\text{poly}}$-reductions.

**Lemma 5.25.** If $A \in \text{ESPACE}$ is $n \log n$-incompressible by $\leq^p_{\text{poly}}$-reductions, then $\text{P/Poly}^{-1}(A)$ has pspace-measure 0.

**Proof.** Assume the hypothesis, fix $c \in \mathbb{N}$ such that $A \in \text{DSPACE}(2^{2^n})$, and let

$$X = \{A \mid A \text{ has } \{0,1\}^* \text{ as a } \text{DSPACE}(2^{2^n})/\text{Poly complexity core}\}.$$ 

By Corollary 5.18, every $\text{DSPACE}(2^{2^n})/\text{Poly}$ complexity core of every language in $\text{P/Poly}^{-1}(A)$ has a dense complement. It follows that $\text{P/Poly}^{-1}(A) \subseteq X^c$. Since $X$ has pspace-measure 1 as a result of Corollary 5.7, it follows that $\text{P/Poly}^{-1}(A)$ has pspace-measure 0. 

We now use Lemma 5.25 to show that the small span phenomenon occurs in $\text{ESPACE}$.

**Theorem 5.26 (Small Span Theorem).** For every $A \in \text{ESPACE}$,

$$\mu(\text{P/Poly}_m(A) \mid \text{ESPACE}) = 0$$

or

$$\mu_{\text{pspace}}(\text{P/Poly}^{-1}(A)) = \mu(\text{P/Poly}^{-1}(A) \mid \text{ESPACE}) = 0.$$
Proof. Let $A \in \text{ESPACE}$ and let $X$ be the set of all languages that are $n^{\log n}$-incompressible by $\leq_p^{\text{poly}}$-reductions. We have two cases.

Case I. If $P/\text{Poly}_m(A) \cap \text{ESPACE} \cap X = \emptyset$, then Theorem 5.8 tells us that
$$P/\text{Poly}_m(A) \mid \text{ESPACE} = 0.$$ 

Case II. If $P/\text{Poly}_m(A) \cap \text{ESPACE} \cap X \neq \emptyset$, then fix a language $B \in P/\text{Poly}_m(A) \cap \text{ESPACE} \cap X$. Since $B \in \text{ESPACE} \cap X$, Lemma 5.25 tells us that
$$\mu_{\text{pspace}}(P/\text{Poly}_m^{-1}(B)) = \mu(P/\text{Poly}_m^{-1}(B) \mid \text{ESPACE}) = 0.$$ 
Since $P/\text{Poly}_m^{-1}(A) \subseteq P/\text{Poly}_m^{-1}(B)$, it follows that
$$\mu_{\text{pspace}}(P/\text{Poly}_m^{-1}(A)) = \mu(P/\text{Poly}_m^{-1}(A) \mid \text{ESPACE}) = 0.$$ 

The small span theorem has several immediate consequences. Most notably, Theorem 5.26 implies that every $P/\text{Poly}$ many-one degree has measure 0 in $\text{ESPACE}$.

Theorem 5.27. For all $A \subseteq \{0,1\}^*$,
$$\mu(\text{deg}^{P/\text{poly}}_m(A) \mid \text{ESPACE}) = 0.$$ 

Proof. Let $A \subseteq \{0,1\}^*$. If $\text{deg}^{P/\text{poly}}_m(A) \cap \text{ESPACE} = \emptyset$, then it is clear that
$$\mu(\text{deg}^{P/\text{poly}}_m(A) \mid \text{ESPACE}) = 0.$$ 
Otherwise, if $\text{deg}^{P/\text{poly}}_m(A) \cap \text{ESPACE} \neq \emptyset$, then fix a $B \in \text{deg}^{P/\text{poly}}_m(A) \cap \text{ESPACE}$. By Theorem 5.26, either
$$\mu(\text{deg}^{P/\text{poly}}_m(B) \mid \text{ESPACE}) = \mu(P/\text{Poly}_m^{-1}(B) \mid \text{ESPACE}) = 0.$$
or

\[ \mu(\deg_{m}^{p/poly}(B) \mid \text{ESPACE}) = \mu(\text{P/Poly}_{m}(B) \mid \text{ESPACE}) = 0. \]

Since \( \deg_{m}^{p/poly}(B) = \deg_{m}^{p/poly}(A) \), it follows that

\[ \mu(\deg_{m}^{p/poly}(B) \mid \text{ESPACE}) = 0. \]

\[ \square \]

We conclude this chapter by mentioning an additional result that follows immediately from Theorem 5.27. Notice that \( \deg_{m}^{p/poly}(\emptyset) = \text{P/Poly} \), so it follows that \( \text{P/Poly} \) has measure 0 in \( \text{ESPACE} \). This fact was originally proven by Lutz in [63].
CHAPTER 6. WEAKLY COMPLETE PROBLEMS

In Chapters 4 and 5 above, we prove tight lower bounds on the complexity of hard and weakly hard problems for the classes $E$, $E_2$, $E_{2SPACE}$, and $E_{2SPACE}$. Similarly, we prove tight upper bounds on the complexity of hard problems for these classes. Note the obvious asymmetry in our results. We show upper and lower bounds on the complexity of hard problems, yet we show only lower bounds on the complexity of weakly hard problems. A natural question to ask is whether or not the same upper bounds apply to the weakly hard problems. In this chapter we answer this question and provide a rigorous explanation for the asymmetry in our results; weakly hard problems do not obey the same upper bounds as hard problems.

In this chapter we establish a strong separation between the notions of weak hardness and hardness by showing that weakly hard problems are not scarce. The fact that weakly hard problems do not obey the same upper bounds as hard problems is seen as a direct consequence of the non-scarcity of weakly hard problems. (Note that if the notions of hardness and weak hardness were equivalent then the upper bounds established in Chapters 4 and 5 would immediately apply to the weakly hard problems.) Let $C$ be any of the classes $E$, $E_2$, $E_{SPACE}$, or $E_{2SPACE}$. The main result of this chapter, Theorem 6.4, shows that the set of weakly $\leq_{m}^{P}$-complete problems for $C$ does not have measure 0 in $C$. This result, in combination with the almost
everywhere lower bounds of Chapters 4 and 5, immediately establishes the existence of weakly $\leq_m^P$-complete problems that do not satisfy the upper bounds previously established for $\leq_m^P$-complete problems. For example, Berman [13] shows that every $\leq_m^P$-complete problem for E has an infinite, polynomial-time decidable subset. In contrast, we show in Corollary 6.7 that there exists a weakly $\leq_m^P$-complete problem for E with no infinite polynomial-time decidable subset. (Note that Corollary 6.7 is implicit in the work of Lutz [65].)

Since the set of $\leq_m^P$-complete problems for $C$ has measure 0 in $C$, our main result also implies that the set of problems that are weakly $\leq_m^P$-complete, but not $\leq_m^P$-complete, for $C$ does not have measure 0 in $C$. This corollary extends the following recent result of Lutz.

**Theorem 6.1** (Lutz [65]). There exist problems that are weakly $\leq_m^P$-complete, but not $\leq_m^P$-complete, for E.

The original proof of Theorem 6.1 required a sophisticated martingale diagonalization argument. Here we simplify and extend this argument to prove the main result. In section 6.1, we review Lutz's original martingale diagonalization argument and present an alternate proof of Theorem 6.1. In section 6.2, we use our simplified martingale diagonalization technique to prove the main result.

### 6.1 Martingale Diagonalization

Stated simply, the goal of martingale diagonalization is to produce languages that "defeat" specific martingales. The basic technique is best illustrated by example. Let $d$ be a martingale and define a language $H_d \subseteq \{0, 1\}^*$ so that the membership of each
string \( s_n \) in \( H_d \) satisfies

\[
[s_n \in H_d] = [d(\chi_{H_d}[0..n - 1])1] \leq d(\chi_{H_d}[0..n - 1]0)].
\]

(Recall from Chapter 2 that \( s_n \) is the \( n^{th} \) element in the standard ordering on \( \{0,1\}^* \)
and \( \chi_{H_d} \) is the characteristic sequence of \( H_d \).) Then the language \( H_d \) "defeats" the
martingale \( d \) in the sense that \( H_d \not\in S^\infty[d] \). To see this, notice that the averaging
condition on \( d \) and the definition of \( H_d \) ensures that

\[
d(\chi_{H_d}[0..n]) \leq d(\chi_{H_d}[0..n - 1])
\]

for every \( n \in \mathbb{N} \) and thus that

\[
\limsup_{n \to \infty} d(\chi_{H_d}[0..n - 1]) \leq d(\lambda) < \infty.
\]

The original proof of Theorem 6.1 uses a heavily modified version of the basic
martingale diagonalization technique to construct a sequence of languages \( H, H_0, H_1, \ldots \)
such that

1. For each \( i \in \mathbb{N} \), \( H_i \in \Sigma \) and \( H_i \leq_m H \).
2. For every \( p \)-computable martingale \( d \), there exists an \( i \in \mathbb{N} \) such that \( H_i \not\in S^\infty[d] \).
3. \( H \in \Sigma \) and is incompressible by \( \leq_m^{\text{DTIME}(2^{\text{TIME}})} \)-reductions.

In the construction, conditions (1) and (2) guarantee that \( P_m(H) \cap \Sigma \not\in S^\infty[d] \) for
every \( p \)-computable martingale \( d \). It follows that \( \mu(P_m(H)|\Sigma) \neq 0 \) and thus that \( H \)
is weakly \( \leq_m^p \)-hard for \( \Sigma \). Condition (3) guarantees that \( H \) is not \( \leq_m^p \)-hard for \( \Sigma \). (By
Theorem 4.19, no \( \leq_m^p \)-hard problem for \( \Sigma \) is incompressible by such reductions.) To
complete the proof of Theorem 6.1, the language $H$ is padded to produce a $C \in E$ with the desired properties.

The proof of Theorem 6.1 uses an involved argument to show that the conditions (1)-(3) can be satisfied simultaneously. This argument hinges on the fact that the set of all $p$-computable martingales can be efficiently enumerated. In [65], such an enumeration is referred to as a rigid enumeration.

**Theorem 6.2** (Martingale Enumeration Theorem [65]). There exists an enumeration $d_0, d_1, \ldots; \hat{d}_0, \hat{d}_1, \ldots$ of all $p$-martingales that satisfy the following three conditions.

(i) $d_0, d_1, \ldots$ is an enumeration of all $p$-martingales.

(ii) For each $k \in \mathbb{N}$, $\hat{d}_k$ is a $p$-computation of $d_k$.

(iii) For all $k, r \in \mathbb{N}$ and $w \in \{0,1\}^*$, $\hat{d}_{k,r}(w)$ is computable in at most $(2 + r + |w|)k^{|k|}$ steps, where $|k| = \log(k + 1)$.

(Lutz's original theorem is stronger, but the above version is sufficient for our purposes.)

Using this enumeration, the original proof of Theorem 6.1 constructs a sequence of languages $H, H_0, H_1, H_2, \ldots$ so that each $H_i$ defeats the $i^{th}$ $p$-computable martingale and so that conditions (1) and (3) are also satisfied. We now use this enumeration in a simplified proof of Theorem 6.1.

The key to our simplified proof of Theorem 6.1 lies in the existence of certain "strong" martingales. Here we say that a martingale is strong if it succeeds on every language that is not weakly $\leq_m^P$-complete.
Definition. Let $C$ be any of the classes $E$, $E_2$, ESPACE, or $E_2$SPACE. Then a martingale $d_s$ is strong for $C$ if every element of $C - S^\infty[d_s]$ is weakly $\leq^P_m$-complete for $C$.

Assume, for the moment, that efficient strong martingales exist. Then we get the following simple proof of Theorem 6.1. Let $d_s$ be strong for $E$, let $d_f$ be a martingale that succeeds on all languages that are not incompressible by $\leq^D_m$-reductions, and let $H_d$ be the language constructed by a basic martingale diagonalization against the martingale $d_s + d_f$. If $H_d \in E$, then we have the following.

(1) $H_d$ is weakly $\leq^P_m$-complete for $E$.

(2) $H_d$ is incompressible by $\leq^D_m$-reductions.

Thus $H_d$ is weakly $\leq^P_m$-complete, but not $\leq^P_m$-complete for $E$ by the argument in [65].

In our simplified proof it is crucial that the basic martingale diagonalization produce a language $H_d \in E$. To ensure this, we must be able to compute the strong martingale $d_s$ efficiently. The following technical lemma guarantees that such efficient strong martingales exist.

Lemma 6.3 (Main Technical Lemma). Let $i, r \in \mathbb{N}$, $w \in \{0, 1\}^*$, $|i| = \log(i+1)$, and $n = \log |w|$. Then there exist a martingale $d_s$ and a computation $\hat{d}_s : \mathbb{N} \times \{0, 1\}^* \rightarrow D$ of $d_s$ such that $d_s$ is strong for $E$ and $\hat{d}_s(w)$ is computable in

$$O\left(n \cdot \sum_{i=0}^{n} (r + \log n + |w|^{\frac{i}{i+1}})^i\right)$$

steps.
Proof. We construct a martingale $d_*$ and a computation $\hat{d}_*$ of $d_*$ such that $d_*$ is strong for $E$ and $\hat{d}_*$ is computable in the stated bound. First some specific notation is necessary.

Let $\langle , \rangle : \{0,1\}^* \times \mathbb{N} \rightarrow \{0,1\}^*$ be the pairing function defined by $\langle x, i \rangle = 1^i01^{\lceil \log_2 |x| \rceil}0x$. For each $i \in \mathbb{N}$, let $f_i : \{0,1\}^* \rightarrow \{0,1\}^*$ be the many-one reduction defined by $f_i(x) = \langle i, x \rangle$. Notice that each $f_i$ is computable in linear time and that $|f_i(x)| = (i + 1)|x| + i + 2$.

For each $i \in \mathbb{N}$ and language $H$, define the language $L_{H,i}$ to be

$$L_{H,i} = f_i^{-1}(H) = \{x \in \{0,1\}^* \mid f_i(x) \in H\}.$$  

We associate initial segments of the characteristic sequence of $L_{H,i}$ with initial segments of the characteristic sequence of $H$ as follows. (Recall from section 2 that the characteristic sequence of a language $H$ is the sequence $\chi_H \in \{0,1\}^\omega$ defined by $\chi_H[i] = \llbracket s_i \in H \rrbracket$.) Define the $i^{th}$ strand of a string $w \in \{0,1\}^*$ to be the substring of $w$ that is mapped to by $f_i$. More precisely, let $w \in \{0,1\}^*$, $b \in \{0,1\}$, and $i \in \mathbb{N}$. Then the $i^{th}$ strand of $w$ is the string $w_{(i)}$ as defined by the following recursion. (In the recursion we write $\#s$ for the position of the string $s$ in the standard enumeration of $\{0,1\}^*$.)

(i) $\lambda_{(i)} = \lambda$.

(ii) $wb_{(i)} = \begin{cases} w_{(i)}b & \text{if } |w| = \#f_i(y) \text{ for some } y \in \{0,1\}^*.
\end{cases}$

Note the following obvious, yet important, properties of strands.

(1) $|w_{(i)}| < |w|^\frac{1}{i+1}$. 

(2) For every \( n \in \mathbb{N} \), there exists \( m_n \in \mathbb{N} \) such that \( m_n \geq n \) and

\[
\chi_{L_{H,i}}[0..n-1] = (\chi_H[0..m_n - 1])_i.
\]

Define, for each \( i \in \mathbb{N} \), the function \( \tilde{d}_i : \{0,1\}^* \rightarrow [0, \infty) \) by

\[
\tilde{d}_i(w) = 2^{-i} \cdot \frac{d_i(w_0) + 1}{d_i(\lambda) + 1},
\]

where \( d_i \) is the \( i^{th} \) martingale in the rigid enumeration of all \( p \)-martingales and \( w \in \{0,1\}^* \). Let \( d_s : \{0,1\}^* \rightarrow [0, \infty) \) be the function defined by

\[
d_s(w) = \sum_{i=0}^{\infty} \tilde{d}_i(w).
\]

It is obvious upon inspection that each of the functions \( \tilde{d}_i \), as well as the function \( d_s \), is a martingale. We show that every language \( H \in E - S^\infty[d_s] \) is weakly \( \leq_m^p \)-complete for \( E \). It follows that \( d_s \) is strong for \( E \).

Let \( H \in E - S^\infty[d_s] \). To see that \( H \) is weakly \( \leq_m^p \)-complete for \( E \), let \( i \in \mathbb{N} \), let \( d_i \) be the \( i^{th} \) martingale in the rigid enumeration of all \( p \)-martingales, and let \( L = L_{H,i} \).

Since \( H \in E \) and \( L \leq_m H \) via \( f_i \), it is clear that \( L \in P_m(H) \cap E \). Moreover, \( L \) is not an element of \( S^\infty[d_i] \). To see this, notice that the second property of strands guarantees that

\[
\limsup_{n \to \infty} 2^{-i} \cdot \frac{d_i(\chi_{L_{H,i}}[0..n-1]) + 1}{d_i(\lambda) + 1} = \limsup_{n \to \infty} \tilde{d}_i(\chi_H[0..m_n - 1]) \\
\leq \limsup_{n \to \infty} \tilde{d}_i(\chi_H[0..n - 1]) \\
\leq \limsup_{n \to \infty} d_s(\chi_H[0..n - 1]).
\]

Since \( H \notin S^\infty[d_s] \), it follows that

\[
\limsup_{n \to \infty} d_i(\chi_L[0..n - 1]) \leq 2^i \cdot (\tilde{d}_i(\lambda) + 1) \cdot \limsup_{n \to \infty} d_s(\chi_H[0..n - 1])
\]
Thus \( P_m(H) \cap E \not\subseteq S^\infty[d_i] \) for every \( i \in \mathbb{N} \). It follows that \( P_m(H) \) does not have measure 0 in \( E \) and that \( H \) is weakly \( \leq^P_m \)-complete for \( E \).

We now define the computation \( \hat{d}_s : \mathbb{N} \times \{0,1\}^* \to \mathbb{D} \) of \( d_s \). Define, for each \( i \in \mathbb{N} \), the computation \( \hat{d}_i^* : \mathbb{N} \times \{0,1\}^* \to \mathbb{D} \) of \( \hat{d}_i \) to be

\[
\hat{d}_i^*(w) = \begin{cases} 
2^{-i} \cdot \frac{\hat{d}_{i,s}(w(i)) + 1}{\hat{d}_i(\lambda) + 1} & \text{if } w(i) \neq \lambda \\
2^{-i} & \text{otherwise},
\end{cases}
\]

where \( w \in \{0,1\}^* \), \( s = 2r + |w(i)| + 2 \), and \( \hat{d}_i \) is the p-computation of \( d_i \) in the rigid enumeration. The computation \( \hat{d}_s \) is then the function defined by

\[
\hat{d}_s(w) = 2^{-\log|w|} + \sum_{i=0}^{\log|w|} \hat{d}_i^*(w),
\]

where \( w \in \{0,1\}^* \) and \( t = 2r + \log \log |w| + 2 \).

The following technical claims show that \( \hat{d}_i^* \) and \( \hat{d}_s \) are computations of \( d_i \) and \( d_s \), respectively.

**Technical Claim 1.** Let \( w \in \{0,1\}^* \) and \( r \in \mathbb{N} \).

(a) \( |\hat{d}_i(w) - \hat{d}_i^*(w)| \leq 2^{-r} \).

(b) The function \( \hat{d}_i^*(w) \) is computable in \( O((4 + 2r + |w(i)|)|w| + 2) \) steps.

**Proof.** To see (a), fix \( s = 2r + |w(i)| + 2 \) as in the definition of \( \hat{d}_i^* \) and let

\[
a = d_i(w(i)) + 1 \\
b = \hat{d}_{i,s}(w(i)) + 1 \\
c = d_i(\lambda) + 1
\]

< \infty.
Since \(a, b, c, d \geq 1\), \(|a - b| \leq 2^{-s}\), and \(|c - d| \leq 2^{-s}\), it follows that \(|ad - bc| \leq 2^{-2r} + 2^{-r}a + 2^{-r}d\). Moreover, the value of \(a\) is at most \(2^{s-2r-2}c\) because \(d_i(w)\) is at most \(2^{|w|} \cdot d_i(\lambda)\). It follows that

\[
|\hat{d}_i(w) - \tilde{d}_i^*(w)| = 2^{-i} \cdot \left|\frac{a}{c} - \frac{b}{d}\right| \\
\leq \frac{|ad - bc|}{cd} \\
\leq \frac{2^{-s}(2^{-s} + a + d)}{cd} \\
\leq \frac{2^{-s}(2^{-s} + 2^{s-2r-2}c + d)}{cd} \\
\leq 2^{-2s} + 2^{-2r-2} + 2^{-s} \\
\leq 3 \cdot 2^{-2r-2} \leq 2^{-r}
\]

The value \(\tilde{d}_i^*(w)\) is produced by first computing \(\hat{d}_i, s(w(i))\) and \(\hat{d}_i, s(\lambda)\) and then combining the results. This takes \(O((4 + 2r + 2|w(i)|)|w|)\) steps.

**Technical Claim 2.** Let \(w \in \{0, 1\}^*, r \in \mathbb{N}\), and \(n = \log|w|\).

(a) \(|d_s(w) - \hat{d}_{s,r}(w)| \leq 2^{-r}\).

(b) The function \(\hat{d}_{s,r}(w)\) is computable in \(O\left(n \cdot \sum_{i=0}^{n} (r + \log n + |w(i)|)|w|\right)\) steps.

**Proof.** Fix \(t = 2r + \log n + 2\) as in the definition of \(\hat{d}_{s,r}\). To see that \(\hat{d}_{s,r}\) approximates \(d_s\) to \(2^{-r}\), first notice that \(w(i)\) is \(\lambda\) if \(i > n\). This fact implies that the sum

\[
\sum_{i=n+1}^{\infty} \hat{d}_i(w) = \sum_{i=n+1}^{\infty} 2^{-i} \cdot \frac{d_i(\lambda) + 1}{d_i(\lambda) + 1}
\]
is $2^{-n}$. It follows that

$$|d_s(w) - \hat{d}_{s,r}(w)| \leq \sum_{i=0}^{n} |\hat{d}_i(w) - \hat{d}_{s,i}^*(w)|$$

$$\leq (n + 1) \cdot 2^{-t}$$

$$\leq 2^{-(2r+1)} + 2^{-(2r+1)} \leq 2^{-r}.$$

Since $\hat{d}_{s,r}(w)$ is produced by first computing the $n$ values of $\hat{d}_{s,i}^*(w)$ for $i$ ranging from 0 to $n$ and then adding the results, it is clear that $\hat{d}_{s,r}$ is computable in $O\left(\sum_{i=0}^{n} (8 + 4r + 2\log n + 2|w(i)|)^{[n]}\right)$ steps. Straightforward algebraic manipulation gives the $O\left(n \cdot \sum_{i=0}^{n} (r + \log n + |w(i)|)[n]\right)$ upper bound. 

Since $|w(i)| < |w|^{\frac{1}{2+t}}$, Technical Claim 2 shows that $\hat{d}_{s}$ can be computed in the stated bound. This completes the proof. 

We conclude this section with the details of the above-sketched proof of Theorem 6.1.

Alternate Proof of Theorem 6.1. By Theorem 4.3 there exists a p-computable martingale $d_f$ such that $d_f$ succeeds on the set

$$Y = \{A \mid A \text{ is not incompressible by } \leq_{m}^{\text{DTIME}(2^{n})}.\text{reductions}\}.$$ 

Fix one such $d_f$ and let $\hat{d}_f$ be a p-computation of $d_f$. Let $d_s$ and $\hat{d}_s$ be the martingale and computation, respectively, from Lemma 6.3. We construct a language $H$ such that $H \in E - S^\infty[d_s + d_f]$. Since $S^\infty[d_s] \subseteq S^\infty[d_s + d_f]$, Lemma 6.3 guarantees that $H$ is weakly $\leq_{m}^{P}$-complete for E. Since $S^\infty[d_f] \nsubseteq S^\infty[d_s + d_f]$, $H \notin S^\infty[d_f]$ and so
$H$ must be incompressible by $\leq^D_{\text{TIME}(2^n)}$-reductions. From Theorem 4.19, it follows that $H$ is not $\leq^P_m$-complete for $E$.

We construct the $H$ via a straightforward martingale diagonalization. Let $y_n = \chi_H[0..n-1]$. Then the membership of $s_n \in \{0,1\}^*$ in $H$ is defined by

$$[s_n \in H_k] = [\hat{d}_{s,2\log n+2}(y_n1) + \hat{d}_{f,2\log n+2}(y_n1) \leq (\hat{d}_{s,2\log n+2}(y_n0) + \hat{d}_{f,2\log n+2}(y_n0)).$$

Notice that

$$(d_s + d_k)(y_{n+1}) \leq (d_s + d_f)(y_n) + \frac{1}{n^2}$$

for each $n \in \mathbb{N}$. It follows immediately that

$$\limsup_{n \to \infty} (d_s + d_f)(y_n) \leq (d_s + d_f)(\lambda) + \sum_{i=1}^{\infty} \frac{1}{i^2} < \infty,$$

and thus that $H \notin S^\infty[d_s + d_f]$.

To see that $H \in E$, let $x = s_m$, let $|x| = n$, and let $k$ be an integer such that $\hat{d}_{f,r}(w)$ is computable in $O((r + |w|)^k)$ steps. The membership of $x \in H$ is decided by (1) computing $y_m$ and (2) computing $\hat{d}_{s,2\log n+2}(y_m1)$, $\hat{d}_{f,2\log n+2}(y_m1)$, $(\hat{d}_{s,2\log n+2}(y_m0)$, and $\hat{d}_{f,2\log n+2}(y_m0)$. For sufficiently large $n$, step (2) can be performed in $O(n^3 \cdot 2^n + 2^{kn})$ steps. It follows that the membership of $x \in H$ can be decided in $O(2^{(k+2)n})$ steps.

6.2 Weakly Complete Problems are Not Rare

The simplified martingale diagonalization argument of the previous section naturally extends to prove the main result of this chapter, namely, that the set of weakly $\leq^P_m$-complete problems for $E$ (and similar classes) does not have measure 0 in $E$. 

\[\square\]
**Theorem 6.4.** Let $C$ be any of the classes $E$, $E_2$, $SPACE$, or $E_2SPACE$. Then the set

$$W_C = \{ A \mid A \text{ is weakly } \leq_m^P\text{-complete for } C \}$$

does not have measure 0 in $C$, i.e., $\mu(W_C \mid C) \neq 0$.

**Proof.** We give the proof for $C = E$. The proofs for $C = E_2$, $C = SPACE$, and $C = E_2SPACE$ are analogous but require modified versions of Lemma 6.3.

Let $d_0, d_1, \ldots, d_0, d_1, \ldots$ be a rigid enumeration of all $p$-martingales from Theorem 6.2, and let $d_s$ and $d^c$ be the martingale and computation, respectively, from Lemma 6.3. We construct a sequence of languages $H_0, H_1, \ldots$ such that each $H_k \in E - S^\infty[d_s + d_k]$. Since $S^\infty[d_s] \subseteq S^\infty[d_s + d_k]$, Lemma 6.3 guarantees that each $H_k$ is weakly $\leq_m^P\text{-complete for } E$. Since $S^\infty[d_k] \subseteq S^\infty[d_s + d_k]$, $H_k \notin S^\infty[d_k]$ for every $k \in \mathbb{N}$. It follows that $W_E$ does not have measure 0 in $E$.

We construct the $H_k$'s via a straightforward martingale diagonalization. Let $y_n = \chi_{H_k}[0..n - 1]$. Then the membership of $s_n \in \{0,1\}^*$ in $H_k$ is defined by

$$[s_n \in H_k] = [\hat{d}_{s,2\log n+2}(y_n1) + \hat{d}_{k,2\log n+2}(y_n1) \leq \hat{d}_{s,2\log n+2}(y_n0) + \hat{d}_{k,2\log n+2}(y_n0)].$$

Notice that

$$(d_s + d_k)(y_{n+1}) \leq (d_s + d_k)(y_n) + \frac{1}{n^2}$$

for each $n \in \mathbb{N}$. It follows immediately that

$$\limsup_{n \to \infty} (d_s + d_k)(y_n) \leq (d_s + d_k)(\lambda) + \sum_{i=1}^{\infty} \frac{1}{i^2} < \infty,$$

and thus that $H_k \notin S^\infty[d_s + d_k]$.

To see that $H_k \in E$, let $x = s_m$ and let $|x| = n$. The membership of $x \in H_k$ is decided by (1) computing $k(x)$ and (2) computing $\hat{d}_{s,2\log n+2}(y_m1), \hat{d}_{k,2\log n+2}(y_m1)$,
(d_{n,2\log n+2}(y_m), and d_{h,2\log n+2}(y_m)). For sufficiently large n, step (2) can be performed in

\[ O(n^3 \cdot 2^n + 2^{2n}) \]

steps. It follows that the membership of \( x \in H_k \) can be decided in \( O(2^{(k+2)n}) \) steps.

\[ \square \]

Theorem 6.4 has a number of immediate corollaries. The first says that the set of problems that are weakly \( \leq_{m}^{P} \)-complete, but not \( \leq_{m}^{P} \)-complete, for E does not have measure 0 in E. This result extends Theorem 6.1.

**Corollary 6.5.** Let \( C \) be any of the classes E, E_2, ESPACE, or E_2SPACE. Then the set

\[ W_{C}^{'} = \{ A \mid A \text{ is weakly } \leq_{m}^{P} \text{-complete, but not } \leq_{m}^{P} \text{-complete, for } C \} \]

does not have measure 0 in \( C \).

**Proof.** Again, we prove the corollary for \( C = E \). The proofs for \( C = E_2, C = ESPACE, \) and \( C = E_2SPACE \) are analogous.

By Theorem 4.14, the set

\[ H_{E} = \{ A \mid A \text{ is } \leq_{m}^{P} \text{-complete for } E \} \]

has measure 0 in E. Since the collection of sets of measure 0 in E is closed under union [63], it follows from Theorem 6.4 that \( W_{C}^{'} \) does not have measure 0 in E. \( \square \)

The next immediate corollary of Theorem 6.4 is that there exist languages that are weakly \( \leq_{m}^{P} \)-complete but not \( \leq_{m}^{P/\text{poly}} \)-complete for ESPACE. This result does not follow from [65].
Corollary 6.6. There exist languages that are weakly $\leq_{\text{me}}^{P}$-complete, but not $\leq_{\text{me}}^{p/\text{poly}}$-complete, for ESPACE.

Proof. By Theorem 5.24, the set

$$ H_{\text{ESPACE}} = \{ A \mid A \text{ is } \leq_{\text{me}}^{p/\text{poly}}\text{-complete for ESPACE} \} $$

has measure 0 in ESPACE. Since the collection of sets of measure 0 in ESPACE is closed under union [63], it follows from Theorem 6.4 that $W_{\text{ESPACE}} - H_{\text{ESPACE}}$ does not have measure 0 in ESPACE and hence is not empty. □

The remaining immediate corollaries of Theorem 6.4 show that previously established upper bounds on the complexity of complete problems do not apply to all weakly complete problems. The first of these says that there exist weakly $\leq_{\text{me}}^{P}$-complete languages for E that are P-bi-immune. (A language $B$ is P-immune if $B$ has no infinite polynomial-time decidable subset. $B$ is P-bi-immune if both $B$ and $B^{c}$ are P-immune.) Previously, Berman [13] established that no $\leq_{\text{me}}^{P}$-complete language for E is P-immune.

Corollary 6.7 (Lutz [65]). There exists a language $H$ that is P-bi-immune and weakly $\leq_{\text{me}}^{P}$-complete for E.

Proof. The set

$$ PB = \{ A \mid A \text{ is P-bi-immune} \} $$

has measure 1 in E by a result of Mayordomo [72]. Since $W_{E}$ does not have measure 0 in E, this implies that $PB \cap W_{E} \cap E \neq \emptyset$. □

In Chapter 4 we show that every DTIME($2^{4n}$) complexity core of every $\leq_{\text{me}}^{P}$-hard language for E has a dense complement. The next corollary of Theorem 6.4
demonstrates the existence of weakly $\leq_{m}^{P}$-complete languages that have $\{0,1\}^{*}$ as a DTIME($2^{4n}$) complexity core.

**Corollary 6.8** (Lutz [65]). There exists a weakly $\leq_{m}^{P}$-complete language $H$ for $E$ that has $\{0,1\}^{*}$ as a DTIME($2^{4n}$) complexity core.

**Proof.** The set

$$BC = \{ A \subseteq \{0,1\}^{*} \mid A \text{ has } \{0,1\}^{*} \text{ as a DTIME}(2^{4n})\text{-complexity core} \}$$

has measure 1 in $E$ by Corollary 4.6. Since $W_{E}$ does not have measure 0 in $E$, this implies that $BC \cap W_{E} \cap E \neq \emptyset$. □

In Chapter 5, we show that for every $\leq_{m}^{p/poly}$-hard language for ESPACE there exists an $\varepsilon > 0$ such that

$$KS^{2^{3n}}(H_{=n}) < 2^{n} - 2^{n^{\varepsilon}} \text{ i.o.}$$

Theorem 6.4 implies that there are weakly $\leq_{m}^{P}$-complete languages for ESPACE that do not satisfy this upper bound.

**Corollary 6.9.** There exists a weakly $\leq_{m}^{P}$-complete language $H$ for ESPACE that satisfies

$$KS^{2^{3n}}(H_{=n}) > 2^{n} - n \text{ a.e.}$$

**Proof.** The set

$$X = \{ A \subseteq \{0,1\}^{*} \mid KS^{2^{3n}}(A_{=n}) > 2^{n} - n \text{ a.e.} \}$$

has measure 1 in ESPACE by Corollary 5.5. Since $W_{\text{ESPACE}}$ does not have measure 0 in ESPACE, this implies that $X \cap W_{\text{ESPACE}} \cap \text{ESPACE} \neq \emptyset$. □
We conclude this chapter by examining the distribution of weakly $\leq^p_m$-hard problems for $E$ inside of $E_2$. We first note that no weakly $\leq^p_m$-hard problem for $E$ can be p-random. (Recall from Chapter 2 that $H$ is p-random if no p-computable martingale succeeds on $H$.)

**Theorem 6.10.** No weakly $\leq^p_m$-hard language for $E$ is p-random.

**Proof.** Let $H$ be weakly $\leq^p_m$-hard for $E$ and let

$$I = \{A \subseteq \{0,1\}^* | A \text{ is incompressible by } \leq^p_m\text{-reductions}\}.$$  

Notice that the set $I$ has measure 1 in $E$ by Theorem 4.3, and the set $P_m(H)$ does not have measure 0 in $E$. It follows that $I \cap P_m(H) \cap E \neq \emptyset$.

Fix $A$ in $I \cap P_m(H) \cap E$. Lemma 4.13 says that if $A$ is in $I \cap E$, then $P_m^{-1}(A) = \{B | A \leq^p_m B\}$ has p-measure 0. Since $H \in P_m^{-1}(A)$, it follows that $H$ is not p-random.

Since the set of p-random languages has measure 1 in $E_2$ by Theorem 2.5, it follows that weakly $\leq^p_m$-hard languages for $E$ are rare in $E_2$.

**Corollary 6.11.** The set

$$WH_E = \{A \subseteq \{0,1\}^* | A \text{ is weakly } \leq^p_m\text{-hard for } E\}$$

has measure 0 in $E_2$.  

It follows from Theorem 6.4 and Corollary 6.11 that there are languages that are weakly $\leq^p_m$-hard for $E_2$ but not weakly $\leq^p_m$-hard for $E$. Surprisingly, this says there exist languages $H$ such that $P_m(H)$ is "not small" inside of $E_2$ but is "small" inside of $E$!
Note that analogous versions of Theorem 6.10 and Corollary 6.11 hold for ES-PACE and $E_2$SPACE.
CHAPTER 7. CONCLUSION

As we mention in Chapter 1, the main results of this dissertation are of the following three general types.

(1) Useful problems contain highly organized information.

(2) Very useful problems are so highly organized that they are unusually simple and hence rare.

(3) Useful problems are, as a whole, not rare and thus are not necessarily simple.

A result of type (1) is given in Chapter 3. There we show that every weakly useful sequence is strongly deep. Since Bennett’s notion of computational depth appears to capture the degree of organization in a given piece of information, an interpretation of this result is that the information contained in weakly useful sequences is highly organized.

Results of type (2) and (3) are given in Chapters 4 and 5. There we investigate the complexity and distribution of $\leq^p_m$-complete languages for E and $\leq^{p/poly}_m$-complete languages for ESPACE. Since these complete problems can be used to efficiently solve every problem in their respective classes, we say that these problems are very useful. We show that the $\leq^p_m$-complete problems for E and the $\leq^{p/poly}_m$-complete problems for ESPACE obey tight upper bounds on their complexity. Moreover, we
show that almost every language in E and ESPACE obeys strong lower bounds on their complexity and that these lower bounds are much higher than the upper bounds for the complete problems. Thus the sets of complete problems for E and ESPACE form “small” subsets of these classes. These results say that the complete problems for E and ESPACE are unusually simple and hence rare.

A result of type (3) is given in Chapter 6. There we investigate the distribution of the weakly \( \leq^P_m \)-complete problems for E and ESPACE. Since these problems can be used to efficiently solve every problem in a non-negligible subset of their respective classes, we say that these problems are useful. We show that the sets of weakly \( \leq^P_m \)-complete problems for E and ESPACE are not “small” subsets of their respective classes. This result and the almost everywhere lower bounds from Chapter 4 and 5 establish the existence of weakly \( \leq^P_m \)-complete problems that do not obey the upper bounds previously established for \( \leq^P_m \)-complete problems. Thus the weakly complete problems are not rare and not necessarily simple.

The results of this dissertation leave several questions unresolved. In Chapter 3 we show that every weakly useful sequence is strongly deep. A natural question to ask is whether or not every strongly deep sequence is weakly useful. Although these notions appear to be very different, it is not immediate that they are different. However, it is reasonable to conjecture that there is a strongly deep sequence that is not weakly useful.

In Chapter 4 we show that the set of \( \leq^P_m \)-hard problems for E has p-measure 0 and hence measure 0 in E. A natural question to ask is whether or not this result holds with “\( \leq^P_m \)” replaced with other reducibilities, such as “\( \leq^P_{tt} \)” or \( \leq^P_T \). This question remains unresolved and may be difficult, especially for the case “\( \leq^P_T \).” Notice that Bennett
and Gill[12] shows that $P^{-1}(A)$ has Lebesgue measure 1 if and only if $A \in BPP$. Thus we cannot extend this result to $\leq P_1$ without showing that $BPP \not\subseteq E$.

Perhaps the most important question this work leaves unresolved is whether or not there exist "natural" problems that are weakly $\leq P_1$-complete, but not $\leq P_1$-complete for $E$. In [65], Lutz conjectures that SAT, the boolean satisfiability problem, is weakly $\leq P_1$-complete but not $\leq P_1$-complete for $E$. However, resolving Lutz's conjecture appears to be very difficult since it implies $P \neq NP$ as well as many strong results not known to follow from the hypothesis $P \neq NP$ (see [68, 38, 64, 65]).
BIBLIOGRAPHY


[68] Lutz, J. H., and Mayordomo, E. Cook versus Karp-Levin: Separating completeness notions if NP is not small, In Proceedings of the Eleventh Symposium on


