Average Case Computational Complexity Theory

by

Tomoyuki Yamakami

A thesis submitted in conformity with the requirements
for the Degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Average Case Computational Complexity Theory

Doctor of Philosophy, 1997

Tomoyuki Yamakami

Graduate Department of Computer Science

University of Toronto

Abstract

The hardest problems in the complexity class **NP** are called **NP**-complete. However, not all **NP**-complete problems are equally hard to solve from the average point of view. For example, the Hamiltonian circuit problem has been shown to be solvable deterministically in polynomial time on the average, whereas the bounded tiling problem still remains hard to solve even on the average. We therefore need a thorough analysis of the average behavior of algorithms.

In response to this need, L. Levin initiated in 1984 a theory of average-case **NP**-completeness. Levin's theory deals with average-case **NP**-complete problems using polynomial-time many-one reductions. The reducibility is a method by which we can classify the distributional **NP** problems.

In this thesis, we develop a more general theory of average-case complexity to determine the relative complexity of all natural average-case intractable problems. We investigate structure of reducibilities, including a bounded-error probabilistic truth-table reducibility. We introduce a variety of relativizations of fundamental average-case complexity classes of distributional decision problems. These relativizations are essential when we attempt to expand our notion of average polynomial-time computability to develop a hierarchy above average **NP** problems.

Average-case analyses are very sensitive to the choice of probability distributions. We have observed that if the input probability distribution decays exponentially with size, for instance, all **NP**-complete problems are solved "fast" on the average. This phenomenon does not reflect a significant feature of average-case analysis. This thesis includes a thorough analysis of structural properties of feasibly computable distributions and feasibly samplable distributions.

In addition, one may ask how we can extract the essential average behavior of algorithms independent of the choice of probability distributions. To answer this question, this thesis introduces the new notion of quintessential computability, which expands the boundary of worst-case feasible computability (such as polynomial-time computability), and asserts the invariance of average-case complexity of algorithms regardless of which feasibly computable distributions are chosen. This thesis examines the hardness of this real computability and its structural properties.

Preface

The theory of average-case **NP**-completeness came forcibly to my attention while I was a visiting scholar at the Universität Ulm from April to August of 1991. In June of 1991, the annual meeting of complexity theorists from the Universität Ulm and the Universitat Politècnica de Catalunya was held in Barcelona. Uwe Schöning, the director of the Abteilung Theoretische Informatik of the Universität Ulm, assigned to young researchers the topics that would be extensively studied at that year's meeting: average-case **NP**-complete problems and local search problems. Six years before, L. Levin had presented his idea of average-case **NP**-completeness, and several important studies were done along these lines.

I started reading these papers and technical reports and enjoyed discussing Levin's definition of "polynomial on average" with Rainer Schuler, who was finishing his thesis on probabilistic computations. The foundations of this thesis were established during this time, and the results were presented at a conference in New Delhi in December, 1992.

In June of 1994, I met Rainer Schuler again at a conference held in Amsterdam. He had with him a paper which solved a problem we had left open in our 1992 paper. We soon started working together, refining his key algorithm to construct hard sets which cannot be computable in feasible time. These results were later presented at a conference in Xi'an, China, in August of 1995 and are also included in this thesis.

This thesis demands of little preparatory knowledge in the theory of computational complexity. Most concepts are thoroughly defined in each section of this thesis or are self-explanatory.

I am extremely grateful to Stephen A. Cook for his hospitality and expert supervision. I thank him also for his direction and support, without which I could not have come to Canada to pursue my Ph.D. degree. My thanks also go to my friend Rainer Schuler who has been my collaborator since I visited the Abteilung Theoretische Informatik of the Universität Ulm in 1991. I would like to thank Jie Wang and Osamu Watanabe for helpful comments and fruitful criticism. Special thanks go to Yuri Gurevich and Alan Selman for his kindness and support. I am also indebted to Leonid Levin and Oded Goldreich for helpful comments. I greatly appreciate the input of my thesis committee members, Steve Cook, Allan Borodin, Alasdair Urquhart, Charlie Rackoff, Yuri Gurevich, Anthony J. Bonner, Rudolf Mathon, and Radford Neal.

I thank my friends Brian Nixon and Luis Dissett at the University of Toronto for their kind advice and encouragement. My special thanks also go to Eric Harley and Debby Repka for pointing out typos and grammatical errors in an early manuscript.

My parents, Fujio and Yoshiko, have supported me emotionally and financially during my studies in Toronto. I also thank my grandmother, Nawo, from the bottom of my heart for spiritual guidance. My great appreciation should go to my fiancée Mitsue Nomura who has helped me write this thesis.

Tomoyuki Yamakami

Toronto, Canada
May 7, 1997

Contents

Chapter 1

Introduction

The new concept of the "automatic computing system" (a term coined by von Neumann) was proposed that gave rise to computers in the mid 1940's. After five decades, computers have come to permeate our society; their presence spans the range from wrist watches to weather forecasting satellites orbiting the earth.

The theory of *computational complexity* has emerged as computer technology has advanced, and now we face more difficulties than ever. When a problem is given, we must write a program or construct a circuit to solve it. To minimize the cost of solving the problem, we must presumably determine its complexity. "Complexity" can be measured in various ways, such as "the running time spent by an algorithm," "the memory space used for an algorithm," "the number of basic operations made by an algorithm," "the number of processors used for a circuit," and so on. Here we focus on an algorithmic model of computation: *worst-case* complexity theory deals with the worst behaviors of algorithms, that is, the maximal complexity of algorithms when an adversary chooses "bad" instances. On the contrary, *average-case* complexity theory analyzes algorithms by measuring their complexity *on the average* over all instances.

Traditional average-case analysis of problems has been performed to determine the expected running time or expected tape space of algorithms to solve given problems under circumstances in which each input instance occurs with a certain probability. We have seen that many important problems, such as the traveling salesperson problem and the Hamiltonian circuit problem, are categorized as the hardest to solve among **NP** problems. The hardest problems in **NP** are called **NP**-complete. All **NP**-complete problems share the same worst-case complexity, but they are not of the same average-case complexity. For example, relatively fast-on-average deterministic algorithms have been found for some famous **NP**-complete problems, such as the graph 3-colorability problem, and the Hamiltonian circuit problem, under naturally selected distributions. Although the notion of expected running time/space is simple and intuitive, it has limitations when used as a base of a consistent and coherent theory and does not address the better understanding of the nature of intractability of problems in both a theoretical and practical sense.

In 1984, Leonid Levin [60] presented a one page paper at the Symposium on Theory of Computing, STOC, proposing the novel idea of defining an average-case complexity measure. Levin demonstrated that a randomized version of an **NP**-complete problem, *the randomized bounded tiling problem*, is complete for a

randomized version of **NP**. This terse paper shed light on what average-case analysis should be. Early works of Gurevich [36] and Ben-David, Chor, Goldreich, and Luby [9] expanded Levin's original idea to establish a coherent framework for average-case complexity theory. Since then, numerous investigations have been made.

This thesis tries to establish a general, consistent, and coherent theory of computational average-case complexity and to contribute to its advancement. In particular, this thesis makes an important addition to Levin's theory of average-case **NP**-completeness by defining average-case hierarchies founded on average polynomial-time computable problems, analogous to the polynomial-time hierarchy. In this thesis, we study the structure and properties of those newly defined hierarchies. We also emphasize the investigation of distributions, which is a recent undertaking. The thesis carries out a thorough analysis of computable distributions and samplable distributions. The most innovative part of this thesis is the introduction in Chapter 7 of *quintessential computability* under a given set of distributions and its investigations. This new concept enables us to discuss a wide range of subjects in average-case complexity theory. We use Kolmogorov complexity, resource-bounded measure, and random oracles to understand the true nature of average behaviors of algorithms.

The thesis consists of eight chapters, each of which addresses a separate issue. Specifically, Chapter 2 provides the reader with the foundations of the theory of computational complexity, the fundamental notions and notation, necessary to read the thesis. Most results come from the author's work (in collaboration with R. Schuler) on average-case complexity theory [97, 98, 119], while some new results are appended elsewhere in the thesis. To avoid confusion, results (theorems, lemmas, etc.) with which the author was involved are listed under *Major Contributions* at the beginning of each chapter with careful attribution. More detailed explanations will be found below.

Easy on the Average. A naive idea of capturing the average behavior of a function f is given as follows. For a distribution μ, let us denote by $\hat{\mu}$ the associated *(probability) density function*. The function f is "expected polynomial on μ-average" if there is a positive integer k such that, for almost all n, $\sum_{x:|x|=n} f(x)\hat{\mu}_n(x) \leq n^k$, where μ_n is the conditional distribution of μ defined on the strings of length n. However, as discussed in Section 3.3, this definition has serious deficiencies, such as lacking the closure property under composition and lacking the property of machine-independence. It therefore cannot be the basis for a consistent, fruitful theory. In the average-case setting, we view a decision problem as a pair consisting of a set of instances and an input distribution, called a *distributional (decision) problem* or *randomized (decision) problems*. The intended interpretation is that, for an algorithm which determines whether $x \in A$, each instance x is given to the algorithm with a probability specified by the distribution.

In contrast, Levin [60] called a function f *polynomial on μ-average* if there exists a positive real number δ such that the expectation $\sum_{x:|x|>0} |x|^{-1} f(x)^{\delta}\hat{\mu}(x)$ converges. This expectation is taken over the infinite set of all nonempty strings. Later Impagliazzo [43] pointed out that we can replace Levin's infinite expectation with a series of finite expectations, on an input ensemble $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$, $\sum_{x:0<|x|\leq n} f(x)^{\delta}\hat{\mu}_{\leq n}(x)$ being bounded by $O(n)$, where $\mu_{\leq n}$ is the conditional distribution of μ on the strings of length at most n. In other words,

it is sufficient to check the expectation over all strings of length at most n.

An intuitive characterization of Levin's notion of "polynomial on μ-average" is given by Schapire [88] as follows: there exists a polynomial p such that, for every positive real number r, $\hat{\mu}(\{x \mid f(x) > p(r \cdot |x|)\}) < 1/r$. Here we remark that $\hat{\mu}$ can be replaced by $\hat{\mu}_{\leq n}$. Based on Schapire's formulation, we are able to extend Levin's *polynomial on μ-average* to *t on μ-average* for an arbitrary function t. Naturally, a distributional decision problem (A, μ) is identified as *"easy" on average* if the problem A is computed by a deterministic Turing machine which halts in polynomial time on μ-average. The collection of all such easy-on-average problems is considered an average-case version of $\mathbf{P}$ by many researchers and is denoted in this thesis by Aver$(\mathbf{P}, *)$ (by AP, AvP, AverP, Aver-P, or Average-P, elsewhere). This class is fundamental to Levin's theory of average-case **NP**-completeness. More generally, we can restrict ourselves to an arbitrary set $\mathcal{F}$ of distributions, and the notation Aver$(\mathbf{P}, \mathcal{F})$ denotes the collection of all easy-on-average distributional problems (A, μ), where μ is taken from $\mathcal{F}$. Under some natural distributions, several **NP**-complete problems are solvable "fast on the average." For example, the "satisfiability problem" [25], the "graph 3-colorability problem" [117], and the "Hamiltonian circuit problem" with edge probability $\frac{1}{2}$ [13] are found to be in Aver$(\mathbf{P}, *)$ under some reasonable distributions.

On the other hand, an average-case counterpart of the class **NP** is the collection of all distributional problems which are pairs of an **NP** set and a *feasibly* computable distribution. This collection is denoted in this thesis by Dist$(\mathbf{NP}, \mathbf{P}$-comp) (by Dist-NP, RNP, or Random-NP elsewhere). Levin raised an intriguing question: " Can all problems in Dist$(\mathbf{NP}, \mathbf{P}$-comp) really be "easy" on the average ?" Ben-David, Chor, Goldreich, and Luby [9] gave the following answer: this is the case unless the nondeterministic linear exponential-time class equals its deterministic counterpart. This thesis is motivated by Levin's open question. Chapter 3 is devoted exclusively to introducing Levin's theory of average-case complexity and its generalization.

To deal with the complexity issue, we generalize the above two classes and introduce the notion Dist$(\mathcal{C}, \mathcal{F})$, which is the collection of all pairs (A, μ), where $A \in \mathcal{C}$ and $\mu \in \mathcal{F}$, and the other four fundamental notions Aver$(\mathbf{NP}, \mathcal{F})$, Aver$(\mathbf{BPP}, \mathcal{F})$, Aver$(\mathbf{RP}, \mathcal{F})$, and Aver$(\mathbf{PSPACE}, \mathcal{F})$.

Input Distributions. Here we would like to remind the reader that average-case analyses are sensitive to the choice of distributions, because "average polynomial-time computability" is founded on the behavior of the distributions in question. The study of distributions is therefore crucial in average-case complexity theory. In Chapter 4, we discuss the complexity of feasible distributions. In particular, we shall focus on two types of distributions: polynomial-time computable distributions and polynomial-time samplable distributions. Gurevich [36] called a distribution μ *polynomial-time computable* if there exists a deterministic Turing machine M such that $|\mu(x) - M(x, 0^i)| \leq 2^{-i}$ for all nonnegative integers i. Ben-David et al. introduced *polynomial-time samplable* distributions which are generated by randomized algorithms (called sampling algorithms [9] or generators [96]) which run in time polynomial in the length of "outputs." By $\mathbf{P}$-comp and $\mathbf{P}$-samp, we denote the sets of polynomial-time computable and samplable distributions, respectively. In Section 4.5, we shall show that polynomial-time samplable distributions are precisely as hard as **PP** sets to

compute deterministically in polynomial time.

Another important notion in Levin's theory of average-case **NP**-completeness is *domination relations* among distributions. When a distribution μ majorizes another distribution ν within a polynomial factor, we say that μ *polynomially dominates* ν. More precisely, μ polynomially dominates ν if there exists a polynomial p such that $p(|x|) \cdot \hat{\mu}(x) \geq \hat{\nu}(x)$ for all strings x. Polynomial-domination of polynomial-time samplable distributions is closely related to the existence of cryptographic one-way functions. A (cryptographic uniform) one-way function is a function which is easy to compute but hard to invert on most instances and is believed to exist by many researchers. Ben-David *et al.* [9] first found this connection and showed that if such one-way functions exist, then there is a polynomial-time samplable distribution which is not polynomially dominated by any polynomial-time computable distribution. In Section 4.7, we shall show that a much weaker assumption, the existence of **NP** sets that are not nearly-**RP**, is enough to get the same conclusion. Here, a set A is said to be nearly-**RP** if some randomized algorithm computes A on most instances, and it behaves like a one-sided bounded-error probabilistic machine on most instances.

Moreover, if two distributions polynomially dominate each other, we say that both are *polynomially equivalent*. For example, every distribution samplable relative to **BPP** sets in time polynomial in the size of output is polynomially equivalent to some polynomial-time samplable distribution. Under the assumption **P** = **NP**, every polynomial-time computable distribution is polynomially-equivalent to some polynomial-time samplable distribution.

Average-Case Reducibility. Chapter 5 focuses on a variety of average-case reducibilities. For decades, researchers have made great efforts to achieve a better understanding of the structure and properties of *intractable* problems. The term **NP**-*complete* was coined to describe the most intractable **NP** problems, and many interesting **NP**-problems are declared to be **NP**-complete, that is, the hardest problems to solve in polynomial time.

Levin's innovation is the invention of an average-case version of such a completeness notion among distributional decision problems. His notion of completeness is based on worst-case polynomial-time many-one reducibility with an extra condition, the so-called *domination condition* for the reduction function, which guarantees that the reduction maps more likely instances to more likely instances. He showed that the "randomized bounded tiling problem" is complete for Dist(**NP**, **P**-comp) under this type of reduction. Since his proof of completeness, only a dozen distributional problems have been found to be complete for Dist(**NP**, **P**-comp). Typical examples are: the "randomized bounded halting problem" [36], the "randomized bounded Post correspondence problem" [36], and the "randomized word problem for Thue systems" [112] under polynomial-time many-one reductions. We shall discuss the issue of deterministic reductions in Section 5.2.

In Section 5.2, we shall formally introduce the (average) polynomial-time many-one reductions and cultivate their structural properties. Wang and Belanger [112] defined *polynomial-time isomorphism* between two distributional decision problems and showed that all known complete problems for Dist(**NP**, **P**-comp) are indeed *polynomially isomorphic*. Section 5.3 will show that several typical distributional problems are

complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) and also polynomially isomorphic to each other.

Incompleteness results have been achieved by Gurevich [36] and by Wang and Belanger [112]. Gurevich [36] first drew attention to distributions of exponentially-small probability, so-called *flat* distributions, and demonstrated that no flat distribution makes a distributional problem complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) unless **NEXP** collapses to **EXP**. We notice that the distribution used for the randomized bounded tiling problem, for example, is not flat. As Wang and Belanger pointed out, if we restrict ourselves to one-one, polynomially honest reductions, we can drop the assumption $\mathbf{EXP} \neq \mathbf{NEXP}$. We shall show that distributions of another type, so-called *sparse* distributions, which were introduced by Gurevich [36], also do not make any distributional problem complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) unless **NP** collapses to **P**. This incompleteness issue will be discussed in Section 5.4.

Another type of important reduction is "probabilistic" or "randomized" reduction. In 1988, Venkatesan and Levin [106] used "random reductions" to demonstrate the intractability of the randomized graph colorability problem. Later Ben-David, Chor, Goldreich, and Luby [9] introduced two more notions: "randomized many-one reductions" and "randomized Turing reductions." In Section 5.5, we shall introduce an average-case version of bounded-error probabilistic truth-table reducibility. Despite the incompleteness result of flat distributions, we are able to prove that the randomized bounded halting problem with a natural flat distribution is also complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) under these reductions.

Average-Case Hierarchies. In worst-case complexity theory, Meyer-Stockmeyer's polynomial-time hierarchy, $\{\mathbf{\Delta}_k^{\mathrm{p}}, \mathbf{\Sigma}_k^{\mathrm{p}}, \mathbf{\Pi}_k^{\mathrm{p}} \mid k > 0\}$, has played a central role in capturing the magnitude of intractability of given problems. Chapter 6 will discuss a hierarchical issue from the average-case complexity point of view.

The *distributional polynomial-time hierarchy under* $\mathcal{F}$ is an extension of the polynomial-time hierarchy in which $\mathbf{\Delta}_k^{\mathrm{p}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}$ are replaced with Dist($\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F}$) and Dist($\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F}$), respectively. We shall show that each Σ-level of the hierarchy under $\mathbf{P}$-comp, Dist($\mathbf{\Sigma}_k^{\mathrm{p}}, \mathbf{P}$-comp), has complete problems under polynomial-time many-one reductions.

The notion will be introduced of *(polynomial-time Turing) self-reducibility* among distributional decision problems. To determine the membership $x \in ?A$, we recursively produce other instances y which are of smaller size than x, and reduce the question $x \in ?A$ to $y \in ?A$. Since the size of instances becomes smaller, these reductions terminate in polynomially-many steps. In worst-case complexity theory, the satisfiability problem, SAT, is a typical example of self-reducible problems. We shall show that most known distributional problems complete for Dist($\mathbf{\Sigma}_k^{\mathrm{p}}, \mathbf{P}$-comp), $k \geq 1$, are self-reducible. Whether all complete problems for Dist($\mathbf{NP}, \mathbf{P}$-comp) are self-reducible, however, is an open question. As an application of self-reducibility, we shall show that Dist($\mathbf{NP}, \mathbf{P}$-comp) $\subseteq$ Aver($\mathbf{BPP}, *$) if and only if Dist($\mathbf{NP}, \mathbf{P}$-comp) $\subseteq$ Aver($\mathbf{RP}, *$).

In Section 6.4, we shall introduce another average-case analogue of the polynomial-time hierarchy, called the*average polynomial-time hierarchy* under a certain set of distribution, to classify distributional decision problems which are hard for Dist($\mathbf{NP}, \mathbf{P}$-comp). The hierarchy is built above Aver($\mathbf{P}, \mathcal{F}$) and Aver($\mathbf{NP}, \mathcal{F}$) using relativized Turing computation.

The model of alternating Turing machines gives another characterization for the polynomial-time hi-

erarchy. Inspired by this characterization, we shall introduce in Section 6.5 an *average polynomial-time alternation hierarchy* under a set $\mathcal{F}$ of distributions using a model of alternating Turing machines. Interestingly, each level of the average polynomial-time alternating hierarchy is characterized by relativized Turing computability relative to classes in the distributional polynomial-time hierarchy. As a result, in contrast to the worst-case situation, the alternating hierarchy is unlikely to coincide with the average hierarchy in general (of course, depending on the underlying set of distributions).

As an example, we shall locate the probabilistic complexity class $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$ in the average polynomial-time alternation hierarchy.

Quintessential Computability. In Chapter 7, we shall shed light on the collective behavior of distributional decision problems under a certain set of distributions, such as **P**-comp or **P**-samp. This approach is new in average-case complexity theory and helps us investigate average-case complexity classes in terms of worst-case complexity classes. More precisely, we shall focus on a class of sets S, called "real **P** under a set $\mathcal{F}$ of distributions," which extracts the essentials of average-case complexity class $\mathrm{Aver}(\mathbf{P}, *)$ in the sense that, for every distribution μ in $\mathcal{F}$, the distributional problem (S, μ) belongs to $\mathrm{Aver}(\mathbf{P}, *)$. In other words, S is computable by some deterministic Turing machine whose running time is polynomial on μ-average. We are particularly interested in feasible distributions, such as **P**-comp. Let us denote by $\mathbf{P}_{\mathbf{P}\text{-comp}}$ the class "**P** under **P**-comp." We return to Levin's original question, $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \subseteq ?\mathrm{Aver}(\mathbf{P}, *)$. Now his question can simply be rephrased in terms of worst-case complexity classes as: "Is **NP** included in $\mathbf{P}_{\mathbf{P}\text{-comp}}$?" Based on the average polynomial-time hierarchy, we further define real polynomial-time classes, $\{\boldsymbol{\Delta}_k^{\mathrm{p}}{}_{\mathcal{F}}, \boldsymbol{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}}, \boldsymbol{\Pi}_k^{\mathrm{p}}{}_{\mathcal{F}} \mid k > 0\}$, called the *real polynomial-time hierarchy under* $\mathcal{F}$. This hierarchy enables us to generalize Levin's question to any level of the real polynomial-time hierarchy under **P**-comp: "Is $\boldsymbol{\Sigma}_k^{\mathrm{p}}$ included in $\boldsymbol{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$?"

We will show that, for every integer $k > 0$, $\boldsymbol{\Delta}_k^{\mathrm{p}} \subseteq \boldsymbol{\Delta}_k^{\mathrm{p}}{}_{\mathcal{F}} \subseteq \boldsymbol{\Delta}_k^{\mathrm{e}}$ and $\boldsymbol{\Sigma}_k^{\mathrm{p}} \subseteq \boldsymbol{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}} \subseteq \boldsymbol{\Sigma}_k^{\mathrm{e}}$ for any set $\mathcal{F}$ of distributions, where $\boldsymbol{\Delta}_k^{\mathrm{e}}$ is the k-th level of the linear-exponential-time hierarchy; in particular, $\mathbf{P} \subseteq \mathbf{P}_{\mathcal{F}} \subseteq \mathbf{E}$ if $k = 1$. Specifically, let us denote by $\mathbf{P}_{\mathbf{E}\text{-comp}}$ the collection of sets computable in polynomial time on average under every exponential-time computable distribution. Using a notion of complexity cores, we are able to show that $\mathbf{P}_{\mathbf{E}\text{-comp}}$ collapses to **P**. More generally, we are able to prove that $\boldsymbol{\Delta}_k^{\mathrm{p}}{}_{\mathbf{REC}\text{-comp}} = \boldsymbol{\Delta}_k^{\mathrm{p}}$ and $\boldsymbol{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{REC}\text{-comp}} = \boldsymbol{\Sigma}_k^{\mathrm{p}}$ for all $k > 0$.

Section 5.6 will discuss hardness results of the average polynomial-time hierarchy under a set of polynomial-time computable distributions. We have already seen the inclusions $\mathbf{P} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}} \subseteq \mathbf{E}$. In 1995, Schuler [92] showed that both inclusions are truly proper using a diagonalization over polynomial-time computable "semi-distributions." (Later he gave an alternative proof based on Kolmogorov complexity.) We extend his technique and show in Section 7.3.1 an even more provocative consequence: $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathrm{DTIME}(2^{c \cdot n})$ for each fixed constant $c > 0$. This result will be extended to any level of the real polynomial-time hierarchy under **P**-comp.

A similar technique again enables us to show that $\boldsymbol{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ has a hard set that is not in $\boldsymbol{\Delta}_k^{\mathrm{p}}/cn$ for each constant $c > 0$, where $\boldsymbol{\Delta}_k^{\mathrm{p}}/f(n)$ in general is the collection of all sets, each of which can be computed by

some $\mathbf{\Delta}_k^{\mathrm{p}}$-type machine with some *advice function* of length $f(n)$. We note that the class of sets computed by non-uniform polynomial-size circuits is exactly the union of all classes $\mathbf{P}/n^k$, $k > 0$. It does not appear to be simple to improve our result to answer the open question of whether all sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ have polynomial-size circuits. However, Schuler [93] recently proved that if all sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ have polynomial-size circuits, then **EXP** collapses to the second level of the polynomial-time hierarchy. Hence, based on the common belief that **EXP** is different from the polynomial-time hierarchy, it seems unlikely that all sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ have polynomial-size circuits. These issues will be discussed in Section 7.3.2.

Another typical example of intractable sets, discussed in Section 7.4, is **P**-immune and **P**-bi-immune sets. **P**-immune sets are sets that do not contain any infinite **P**-subsets in them, and **P**-bi-immune sets are **P**-immune sets whose complements are also **P**-immune. We show that there are some non-sparse **P**-immune sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$, but $\mathbf{P}_{\mathbf{P}\text{-comp}}$ has no **P**-bi-immune sets. This fact exhibits the structural difference between $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and the class **E**, which has both **P**-immune and **P**-bi-immune sets. Using the fact regarding **P**-bi-immunity, $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is shown to be *small* with respect to Lutz's resource-bounded measure, where a complexity class is often called *small* if it has p-measure 0. (Note that **E** has p-measure 1.) As an immediate consequence, if **NP** is included in $\mathbf{P}_{\mathbf{P}\text{-comp}}$, then **NP** has p-measure 0, and this consequence again contradicts the popular belief that **NP** is not small. Along the same lines, Schuler [94] showed that the truth-table closure of $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and the Turing closure of $\mathbf{P}_{\mathbf{P}\text{-comp}}$ have different measures.

Section 7.5 will show that $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is not closed under polynomial-time many-one reductions, the existential operators, or the probabilistic operators. Hence, the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$, for example, is structurally different from most of the well-known complexity classes, such as **P**, **NP**, **BPP**, and **PP**. However, it is not known whether $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is closed under p-honest many-one reductions. Notice that the class $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-samp}}$, real $\mathbf{\Delta}_k^{\mathrm{p}}$ under **P**-samp, is closed under p-honest polynomial-time many-one distributions. We shall show that if $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is not closed under p-honest polynomial-time many-one reductions, then there is a polynomial-time samplable distribution which is not polynomially dominated by any polynomial-time computable distribution. Under p-honest many-one reductions, we are able to show that there exists a pair of sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ which are not reducible to each other, a so-called *incomparable pair*.

The quintessential complexity class $\mathbf{BPP}_{\mathcal{F}}$ exhibits another structure. Due to Ben-David, Chor, Goldreich, and Luby [9], the assumption $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$ implies the conclusion $\mathbf{\Theta}_2^{\mathrm{p}} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$, where $\mathbf{\Theta}_2^{\mathrm{p}}$ is the class of sets computable in polynomial time with nonadaptive queries to **NP** oracles. On the other hand, Schuler and Watanabe [96] extended a result of Venkatesan and Levin [106] and showed that the $\mathbf{NP} \subseteq ?\mathbf{BPP}_{\mathbf{P}\text{-samp}}$ question is equivalent to the $\mathbf{NP} \subseteq ?\mathbf{BPP}_{\mathbf{P}\text{-comp}}$.

As shown by Ben-David *et al.* [9], $\mathbf{NP} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$ leads to the conclusion $\mathbf{E} = \mathbf{NE}$. On the other hand, $\mathbf{NP} \not\subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$ yields the consequence $\mathbf{P} \neq \mathbf{NP}$. Hence, the $\mathbf{NP} \subseteq ?\mathbf{P}_{\mathbf{P}\text{-comp}}$ question cannot be easily solved in the non-relativized world. At this point, we have no prospect for answering Levin's question either affirmatively or negatively. Now let us turn our interest to a relativization of this question. In 1981, Bennett and Gill [8] introduced a notion of *random oracles* to show that **P** is different from **NP** in "most" relativized worlds. More precisely, if an oracle set is chosen at random, the probability that **P** differs from **NP** relative to this oracle is 1. In Section 7.7, we shall show that **NP** and $\mathbf{P}_{\mathbf{P}\text{-comp}}$ are mutually exclusive

(i.e., $\mathbf{NP} \not\subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$ and $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{NP}$) in "most" relativized worlds. To be more precise, let us denote by $\mathbf{P}^X_{\mathbf{P}^X\text{-comp}}$ a "natural" relativization of the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ relative to oracle X. We will show that, with probability 1, $\mathbf{NP}^X \not\subseteq \mathbf{P}^X_{\mathbf{P}^X\text{-comp}}$ and $\mathbf{P}^X_{\mathbf{P}^X\text{-comp}} \not\subseteq \mathbf{NP}^X$, relative to random oracle X.

Chapter 2

Foundations of Computational Complexity Theory

2.1 Introduction

The theory of computational complexity first drew attention from mathematicians as a weak notion of recursive functions. To measure the complexity of a given problem, we use particular models of computation, such as Turing machines, circuits, or PRAM's to solve the problem.

In this chapter, we shall define and explain most of the fundamental notions and notations in *(worst-case) computational complexity theory* so that the uninitiated reader can read through this thesis without the help of supplementary textbooks.

In Section 2.2, we shall cover the elementary notions of *graphs, numbers, sets*, and *functions*. The basic terminology in *probability theory* and *logic* will be also introduced. The thesis follows the standard terminology often used in mathematics and theoretical computer science.

We use *Turing machines* as a model of computation. In general, deterministic Turing machines compute partial *recursive* functions, but our interests lie only in resource-bounded computations, and we need the notions of *running time* and *tape space* of the Turing machines. The reader should pay careful attention to the models we shall use in this thesis because different models lead to different consequences. Several variations of Turing machines will be introduced in Section 2.3, and many popular complexity classes, such as **P** and **NP**, will be defined in Section 2.5.

The field of *randomized algorithms* has grown tremendously in the last decade and has found many applications because of their simplicity and speed. We shall introduce the basic notions of *randomized Turing machines*, *probabilistic Turing machines*, and *random functions* in Section 2.4.

In Section 2.6, *universal hash functions* will be introduced. Hash functions are a useful tool in designing randomized algorithms.

Section 2.7.1 will explain the theory of *polynomial analysis* initiated by Ko and Friedman [55] in the early

1980's. The theory of *Kolmogorov complexity* also provides us with a succinct description of information. We also cover the notion of *resource-bounded Kolmogorov complexity measure* and Lutz's *resource-bounded measure theory*, which are popular in structural complexity theory.

For complete references, the reader may refer to [42, 91, 4, 45, 80].

Major Contributions. Although this chapter is introductory, a few results are included.

Lemma 2.5.7 offers a new characterization of all Δ-level classes in the polynomial-time hierarchy by the model of semi-deterministic alternating Turing machines which run in polynomial time with constant alternation.

Proposition 2.6.4 shows the existence of an **NP** set which is not nearly-**RP**, provided that strong one-way functions exist.

2.2 Fundamental Notions and Notation

We shall begin with terminology from mathematical logic and then explain mathematical notions and notations: graphs, sets, numbers, strings, and functions. This section will include a preliminary introduction to probability theory.

2.2.1 Logic

In propositional logic, we deal only with *Boolean variables* which take values 1 (*truth*) and 0 (*falsehood*). (Note that traditionally, in mathematical logic, 0 represents "falsehood" and 1 represents "truth.") The *terms* are Boolean variables and the *logical constants* 0 and 1. As *logical connectives*, we use the symbols $\neg$ (negation), $\wedge$ (conjunction), and $\vee$ (disjunction). The set of *(propositional) formulas* is defined by the following clauses:

(i) every term is a formula;

(ii) if α and β are formulas, then $\neg(\alpha)$, $(\alpha \wedge \beta)$, and $(\alpha \vee \beta)$ are formulas; and

(iii) formulas are defined only by clauses (i)-(ii).

Unless there may be confusion, we freely omit parentheses from formulas: e.g., $\neg\neg\alpha$ and $\alpha \wedge (\beta \vee \neg\gamma)$. The negation of a Boolean variable v is sometimes denoted by $\overline{v}$ for simplicity. A Boolean variable and its negation are called *literals*.

Let $\alpha = \alpha(x_1, x_2, \ldots, x_n)$ be a formula with all distinct variables being explicitly exhibited as $x_1, x_2, \ldots, x_n$. We write $Var(\alpha)$ for the set $\{x_1, x_2, \ldots, x_n\}$. A *truth assignment for* α is a function $\sigma : Var(\alpha) \to \{T, F\}$. Given a truth assignment σ, we define an *evaluation* $[\alpha]_\sigma$ of α on σ in the following recursive way:

(i) in the case that α is a variable v, $[\alpha]_\sigma = T$ if and only if $\sigma(v) = T$;

(ii) in the case that α is of the form $\neg(\beta)$, $[\alpha]_\sigma = T$ if and only if $[\beta]_\sigma = F$;

(iii) in the case that α is of the form $(\beta_0 \wedge \beta_1)$, $[\alpha]_\sigma = T$ if and only if $[\beta_0]_\sigma = T$ and $[\beta_1]_\sigma = T$; and

(iv) in the case that α is of the form $(\beta_0 \vee \beta_1)$, $[\alpha]_\sigma = T$ if and only if $[\beta_0]_\sigma = T$ or $[\beta_1]_\sigma = T$.

A propositional formula α is *satisfiable* if there exists a truth assignment σ for α such that $[\alpha]_\sigma = T$. In this case, σ is said to *satisfy* α. For example, the formula $\neg\neg(x \vee y) \wedge (\neg z \vee y)$ is satisfiable, witnessed by the assignment σ such that $\sigma(x) = \sigma(y) = T$ and $\sigma(z) = F$. A formula α is *valid* (or a *tautology*) if $[\alpha]_\sigma = T$ for any truth assignment σ for α.

For a property $\mathcal{Q}$, the notation $\forall x \mathcal{Q}(x)$ means that $\mathcal{Q}(x)$ holds for all elements x, and the notation $\exists x \mathcal{Q}(x)$ means that there exists an element x satisfying $\mathcal{Q}(x)$. The notation $\exists! x \mathcal{Q}(x)$ means that there exists the unique element x satisfying $\mathcal{Q}(x)$. For a property $\mathcal{Q}$ defined on an infinite set S, we say that $\mathcal{Q}(x)$ holds *for almost all* (or *almost every*) x in S if the set $\{x \in S \mid \mathcal{Q}(x) \text{ does not hold }\}$ is finite. In this case, we also say that $\mathcal{Q}$ holds *almost everywhere*. The notation $\overset{\infty}{\forall} x \mathcal{Q}(x)$ means that $\mathcal{Q}(x)$ holds for almost all x, and $\overset{\infty}{\exists} x \mathcal{Q}(x)$ means that $\mathcal{Q}(x)$ holds for infinitely many x. Clearly $\overset{\infty}{\forall}$ and $\overset{\infty}{\exists}$ are dual concepts.

Generally, for a property $\mathcal{Q}$, we write $[\mathcal{Q}] = 1$ if $\mathcal{Q}$ is true, and $[\mathcal{Q}] = 0$ otherwise. For a set S, χ_S denotes the *characteristic function for* S that is defined as $\chi_S(x) = [x \in S]$. (Note that "characteristic functions" here are different from those used in probability theory.) For brevity, we also use the notation $S(x)$ to mean $\chi_S(x)$.

2.2.2 Sets and Numbers

Sets. Intuitively, a *set* is a collection of objects, called its *members* or *elements*. The notation $x \in A$ expresses that x is an element of A, and $\in$ is called the membership relation. The symbol $\emptyset$ denotes the *empty set* that contains no elements. We use the standard set notation $\{\cdot \mid \cdot\}$. For example, the notation $\{x \mid Q(x)\}$ represents the set whose elements x satisfy a property $Q(x)$. For two sets A and B, we say A is a *subset* of B, symbolically $A \subseteq B$, if every element of A is an element of B.

For two sets A and B, the *intersection* of A and B is denoted $A \cap B$ and is defined by $A \cap B = \{a \mid a \in A \wedge b \in B\}$. The *union* of A and B is denoted $A \cup B$ and is defined by $A \cup B = \{a \mid a \in A \vee b \in B\}$. The set $A - B$ denotes the of A and B that is defined by $A - B = \{a \mid a \in A \wedge b \notin B\}$.

The of A and B , denoted $A \times B$, is the set of all ordered pairs (a, b) such that $a \in A$ and $b \in B$, where an *ordered pair* is the set $\{a, \{a, b\}\}$. In contrast, the set $\{a, b\}$ is sometimes referred to as an *unordered pair*. The *power set* of S is denoted by $\mathcal{P}(S)$ and is defined as the collection of all subsets of S, i.e., $\mathcal{P}(S) = \{A \mid A \subseteq S\}$.

For a set S, $\|S\|$ denotes the *cardinality* of S that intuitively expresses the number of elements in S. If S is not finite, then let $\|S\| = \infty$.

Binary Relations. A *binary relation* on a set S is a subset of the Cartesian product $S \times S$, i.e., $\{(a, b) \mid a, b \in S\}$. Conventionally, we write aRb when $(a, b) \in R$. For a binary relation R on S, we say that R is *reflexive* if aRa holds for all elements $a \in S$, and that it is *transitive* if aRb and bRc imply aRc for all $a, b, c \in S$. Moreover, a relation R on S is *symmetric* if aRb implies bRa for all $a, b \in S$; on the other hand, R is *antisymmetric* if aRb and bRa imply $a = b$ for all $a, b \in S$.

A *binary operator* $\oplus$ on a set S is called *associative* if $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ for all $a, b, c \in S$; *commutative* if $a \oplus b = b \oplus a$ for all $a, b \in S$.

Numbers. Let $\mathbb{Z}$ be the set of all *integers* $\{\cdots, -2, -1, 0, 1, 2, \cdots\}$, and let $\mathbb{N}$ denote the set of all nonnegative integers, called *natural numbers*. The set of all *rational numbers* $\{\frac{m}{n} \mid m, n \in \mathbb{Z}, n \neq 0\}$ is simply denoted by $\mathbb{Q}$, and $\mathbb{Q}^+$ denotes the set of all nonnegative rational numbers. Similarly, the notation $\mathbb{R}$ denotes the set of all *real numbers*, and in particular, we denote by $\mathbb{R}^+$ the set of all nonnegative real numbers. (Remember that the superscript $+$ does not mean "positive.") We use the notation ∞ to mean the *infinity*, and let $\mathbb{R}^{+\infty} = \mathbb{R}^+ \cup \{\infty\}$ and $\mathbb{R}^\infty = \mathbb{R} \cup \{\infty, -\infty\}$. For the arithmetical operations $+$ (addition) and $\cdot$ (multiplication), we follow the standard convention: for any numbers $r \in \mathbb{R}$ and $s \in \mathbb{R}^+ - \{0\}$, $r + \infty = \infty + r = \infty$, $s \cdot \infty = \infty \cdot s = \infty$, $-s \cdot \infty = \infty \cdot (-s) = -\infty$, and $0 \cdot \infty = \infty \cdot 0 = 0$. Moreover, we assume that $-\infty < r$ and $r < \infty$ for any real number $r \in \mathbb{R}$.

The *absolute value* of a real number is denoted $|r|$.

For any two real numbers a and b $(a \leq b)$, let (a, b) denote an *open (real) interval* defined by $(a, b) = \{x \in \mathbb{R} \mid a < x < b\}$; let $[a, b)$ and $(a, b]$ be *half-open intervals* which are defined by $[a, b) = (a, b) \cup \{a\}$ and $(a, b] = (a, b) \cup \{b\}$, respectively; and let $[a, b]$ be an *closed interval* defined by $[a, b] = (a, b) \cup \{a, b\}$.

For a real number x, let $\lfloor x \rfloor$ (*floor* of x) be the maximal integer not exceeding x, and let $\lceil x \rceil$ (*ceiling* of x) be the minimal integer not smaller than x.

Lebesgue Measure. For a closed interval $I = [a, b]$ of the line $\mathbb{R}$, let $|I| = b - a$. Let $S = \{I_k\}_{k \in \mathbb{N}}$ be a countable collection of closed intervals on $\mathbb{R}$. For a subset E of $\mathbb{R}$, we say that S is a *covering* of E if $E \subseteq \bigcup_{i=k}^{\infty} I_k$. The *Lebesgue outer measure* of a set E, denoted $\mathbf{m}^*(E)$, is defined by

$$\mathbf{m}^*(E) = \inf \left\{ \sum_{k=0}^{\infty} |I_k| \;\middle|\; \{I_k\}_{k \in \mathbb{N}} \text{ is a covering of } E \right\},$$

where the infimum is taken over all coverings of E. If no such covering exists, then take $\mathbf{m}^*(E) = \infty$.

A set E is called *(Lebesgue) measurable* if, for every set $S \subseteq \mathbb{R}$,

$$\mathbf{m}^*(S \cap E) + \mathbf{m}^*(S - E) = \mathbf{m}^*(S).$$

If E is measurable, its Lebesgue outer measure is called its *Lebesgue measure* (or simply *measure*) and is denoted by $\mathbf{m}(E)$. Note that $\mathbf{m}([0, 1]) = 1$. (Thus, $\mathbf{m}$ is a probability measure on the sample space $[0, 1]$.) It is well known that, assuming the *axiom of choice*, there exists a non-measurable set (see, e.g., [115]).

Polynomials and Logarithms. We are interested only in polynomials and logarithms with integer coefficients. For a positive integer d, a *polynomial (in n) of degree d* is a function $p(n)$ of the form:

$$p(n) = \sum_{i=0}^{d} a_i n^i,$$

where each $a_i \in \mathbb{Z}$ and $a_d \neq 0$. The constants $a_0, a_1, \ldots, a_d$ are called the *coefficients* of the polynomial. *Exponentials* are functions of the form $2^{p(n)}$, where p is some polynomial. In particular, we call a function a *linear-exponential* if it is of the form 2^{cx+d} for some constants $c, d \in \mathbb{Q}^+$.

This thesis uses mainly *logarithms to base 2*, and for the sake of convenience, we often omit the base and simply write $\log x$ for $\log_2 x$. Whenever we deal with logarithms of rational numbers, we follow a special convention: we define $\log z$ to be 0 whenever $0 < z < 1$ to simplify the case-by-case description. For brevity, we also write $\mathrm{llog}(n)$ for $\lfloor \log_2(n+1) \rfloor$ and write $\mathrm{ilog}(n)$ for $\lceil \log_2 n \rceil$ for all $n \in \mathbb{N}$.

The notation $\log^{(k)} n$ denotes k iterations of logarithms, namely, define $\log^{(0)} n = n$, and $\log^{(k)} n = \log(\log^{(k-1)} n)$ for $k \geq 1$. Also let $\log^* n = \min\{k \in \mathbb{N} \mid \log^{(k)} n \leq 1\}$. The function $\log^* n$ grows extremely slowly. For example, $\log^* 16 = 3$ and $\log^* 65536 = 4$.

The kth *Harmonic number*, H_k, is defined by $\sum_{i=1}^{k} \frac{1}{i}$.

The *binomial coefficients* are defined as follows: for $n, k \geq 0$, if $n \geq k$, then,

$$\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n \\ n-k \end{array} \right) = \frac{n!}{k!(n-k)!},$$

and if $k > n$, then $\binom{n}{k} = 0$.

2.2.3 Graphs

A *directed graph* G is a pair (V, E), where V is a finite set and E is a binary relation on V (i.e., a subset of $V \times V$). The set V is called a *vertex set* or *node set*, and its element is called a *node* or *vertex*. The set E is called an *edge set*, and its element is called an *edge*. An *undirected graph* $G = (V, E)$ is a variation of directed graph whose edge set is a symmetric relation. For an undirected graph, we identify two edges (a, b) and (b, a) and often write $\{a, b\}$ as an unordered pair.

We say that a node t is *adjacent* to a node s if (s, t) is an edge in a graph.

A *(finite) path of length* k from a node s to a node t in a graph $G = (V, E)$ is a (finite) sequence $(v_0, v_1, \ldots, v_k)$ of nodes in N such that $s = v_0, t = v_k$, and $(a_i, a_{i+1}) \in E$ for all i with $0 \leq i < k$. In this case, we say that the path p *contains* the nodes $v_0, v_1, \ldots, v_k$ and also the edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$. A node t is *reachable* from a node s if there exists a path p from s to t.

A path is *simple* if all nodes in the path are distinct.

Given a path $p = (v_0, v_1, \ldots, v_k)$, a *subpath* p' is a subsequence of p; that is, for some i, j with $0 \leq i < j \leq k$, $p' = (v_i, v_{i+1}, \ldots, v_j)$.

We can naturally extend the definition of graphs and paths to *infinite graphs* and *infinite paths*. For example, an infinite path from a node s in a graph is an infinite sequence, rather than a finite one, starting from s.

A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Given a set $V' \subseteq V$, the subgraph of $G = (V, E)$ *induced* by V' is the graph $G' = (V', E')$, where $E' = \{(u, v) \in E \mid u, v \in V'\}$.

An undirected graph is *connected* if every two nodes are reachable from each other.

In a graph, a path $p = (v_0, v_1, \ldots, v_k)$ forms a *cycle* if (i) $v_0 = v_k$ and (ii) $v_0 \neq v_i$ for some i with $0 < i < k$. A graph with no cycle is said to be *acyclic*.

A *forest* is an acyclic, undirected graph, and a *tree* is a connected, acyclic, undirected graph. In particular, the tree that contains no nodes is called the *empty tree* or *null tree*.

A *rooted tree* is a tree in which one of the nodes is distinguished from the others; this distinguished node is called the *root* of the tree.

Let x and y be any nodes in a rooted tree $T = (V, E)$ with root r. The node y is called an *ancestor* of x if there exists a path from r to x which contains y. If y is an ancestor of x, then x is a *descendant* of y. (Note that x is an ancestor and descendant of x itself.) The node y is called a *parent* of x if (y, x) is an edge on the path from r to x. If y is a parent of x, then x is a *child* of y. Any two nodes which have the same parent are *siblings*. A node with no children is called a *leaf* (or *external node*), while the other non-leaf nodes are called *internal nodes*.

A *subtree rooted at* x is the tree induced by the set of all descendants of x.

The *degree* of a node x in a rooted tree T is the number of children of x in T. The *depth* of a node x is the length of the path from the root of T to x. The *height* of T is the largest depth of any node in T.

2.2.4 Finite and Infinite Strings

An *alphabet* Σ is a nonempty, finite set. Given an alphabet Σ, a *word* or *string* over Σ is a finite sequence of symbols from Σ. The *empty string* is the unique string consisting of no symbols and is denoted by λ. Let us denote by Σ^* the set of all strings over Σ (of course, Σ^* contains λ), and for the sake of convenience, set Σ^+ to be $\Sigma^* - \{\lambda\}$, the set of all nonempty strings.

In this thesis, however, we consider only the binary alphabet $\Sigma = \{0, 1\}$ (a string over $\{0, 1\}$ is often called a *binary string*) because this restriction does not affect any of our arguments.

The *length* of a string x is the number of symbols in x and is denoted by $|x|$. For example, $|01100| = 5$, and in particular, $|\lambda| = 0$. For every $n \in \mathbb{N}$, let Σ^n ($\Sigma^{\leq n}$, $\Sigma^{\geq n}$, respectively) denote all strings of length n (length $\leq n$, length $\geq n$, respectively). We note that a subset of Σ^* is sometimes called a *language* over Σ. For two strings x and y, the *concatenation* of x and y is the string consisting of the symbols of x followed by the symbols of y, and is denoted by xy (or sometimes $x \cdot y$). For example, if $x = 0110$ and $y = 11011$, then $xy = 011011011$. Given a string s, $s\Sigma^n$ denotes the set $\{sy \mid y \in \Sigma^n\}$. For a string x and a natural number n, the notation x^n is recursively defined by: $x^0 = \lambda$, and $x^{n+1} = x \cdot x^n$ for $n \in \mathbb{N}$.

We assume the *standard order* on Σ^*:

$$\lambda < 0 < 1 < 00 < 01 < 10 < 11 < 000 < 001 < 010 < 011 < 100 < 101 < 110 < 111 < 0000 < \ldots$$

(Sort length-wise and then sort lexicographically.) With respect to this order, x^- denotes the *predecessor* of x if one exists, and x^+ denotes the *successor* of x. For example, $0110^+ = 0111$ and $0110^- = 0101$. This ordering enables us to identify strings with natural numbers in the following fashion: let $s_0 = \lambda$, $s_1 = 0$, $s_2 = 1$, $s_3 = 00$, and so forth. In general, let s_n be the n-th string (N.B. λ is the 0th string) of Σ^* in the order. It is easy to see that $|s_n| = \mathrm{llog}(n)$.

It is convenient to define *infinite strings* as infinite sequences of symbols from Σ. We sometimes call a string in Σ^* a *finite string* to stress the finiteness of strings. For simplicity, Σ^∞ denotes the set of all infinite strings.

We say that x is a *prefix* of y, symbolically $x \sqsubseteq y$, if $xs = y$ for some string s. For a string x and a natural number i with $i \leq |x|$, the notation $x_{\leftarrow i}$ denotes the first i bits of x, i.e., the string s such that $|s| = i$ and $s \sqsubseteq x$. For the sake of convenience, whenever $i > |x|$, set $x_{\leftarrow i} = x$. Furthermore, by $x_{\rightarrow i}$ we mean the string s such that $x = x_{\leftarrow i-1}s$. Hence, $x = x_{\leftarrow i-1}x_{\rightarrow i}$.

Let f be a function on $\mathbb{N}$. For a set $S \subseteq \Sigma^*$, S is *of density* $f(n)$ if $\|S \cap \Sigma^k\| = f(k)$ for all $k \in \mathbb{N}$.

The *complement of* a set A, symbolically $\overline{A}$, is $\Sigma^* - A$, and the *symmetric difference* of two sets A and B, symbolically $A \triangle B$, is $(A - B) \cup (B - A)$. The *disjoint union* of A and B, symbolically $A \oplus B$, is the set $\{0x \mid x \in A\} \cup \{1x \mid x \in B\}$.

Any subset of $\{0\}^*$ or $\{1\}^*$ is called a *tally* set, and TALLY denotes the collection of all tally sets. A set S is *(polynomially) sparse* if there exists a polynomial p such that $\|S \cap \Sigma^n\| \leq p(n)$ for all $n \in \mathbb{N}$. By SPARSE, we denote the collection of all sparse sets. By definition, TALLY $\subseteq$ SPARSE.

Dyadic Rational Numbers. A real number r in the unit interval $[0, 1]$ is uniquely identified with its *shortest* binary representation, i.e., of the form

$$\sum_{i=0}^{m} a_i \cdot 2^i + \sum_{j=1}^{\infty} b_j \cdot 2^{-j},$$

where all a_i's and b_j's are in $\{0, 1\}$ (the term "shortest" is necessary because, for example, the binary representation of the number 0.2 is 0.1 as well as $0.0111 \cdots 1 \cdots$). We use the notation $(a_m \cdots a_0.b_1 \cdots b_k \cdots)_2$ to denote this (finite or infinite) binary representation. This expression helps us identify a real number with a pair of (finite or infinite) strings $a_m \cdots a_0$ and $b_1 \cdots b_k \cdots$ separated by ".", the delimiter symbol. By padding 0's if necessary, we can view r as an infinite string in Σ^∞.

Let us define *dyadic rational numbers* as rational numbers with *finite* binary representations. Here are examples: 9.25 is a dyadic rational number and is identified with the string 1001.01, but 2.3 is not a dyadic rational number because its binary representation is of the form $(10.01001 \cdots)_2$ and is infinite.

2.2.5 Functions

In general, we will be using n-ary (partial) functions. For a function f, $\mathrm{dom}(f)$ (*domain* of f) denotes the set of elements from which f maps, and $\mathrm{ran}(f)$ (*range* of f) denotes the set of elements to which f maps. We say that f is a (partial) function *from* A *to* B (or f maps *from* A *to* B), symbolically $f : A \to B$, if $A = \mathrm{dom}(f)$ and $\mathrm{ran}(f) \subseteq B$, and that f is a (partial) function *on* A if f maps from A to A. A function f is *one-one* (or *injective*) if, for any two elements $x, y \in \mathrm{dom}(f)$, $f(x) = f(y)$ implies $x = y$, and f is called *onto* (or *surjective*) if, for every element $y \in \mathrm{ran}(f)$, there exists an element x such that $f(x) = y$. If a function f is one-one and onto, then we call f a *bijection* (or *bijective*).

For a function f and an element y, in general, the notation $f^{-1}(y)$ (*inverse image* of y by f) denotes the

set $\{x \in \mathrm{dom}(f) \mid f(x) = y\}$; however, if this set is a singleton (i.e., $\|\{x \in \mathrm{dom}(f) \mid f(x) = y\}\| = 1$), then by convention $f^{-1}(y)$ denotes the element x such that $f(x) = y$.

The *lambda notation* in λ calculus is a useful tool for describing functions by their values. Based on each value $f(x)$ of a function f, the lambda notation "$\lambda x.f(x)$" denotes the function f itself. Here we shall see some examples. The notation $\lambda x.(c \log x + d)$ expresses the function f defined as $f(z) = c \log z + d$ for all z, and $\lambda x.2^{cx^k+d}$ expresses the function f defined as $f(z) = 2^{cz^k+d}$ for all z.

For two functions f and g, provided that $\mathrm{ran}(g) \subseteq \mathrm{dom}(f)$, the *composition* $f \circ g$ expresses the function h such that $h(x) = f(g(x))$ for all $x \in \mathrm{dom}(g)$.

We say that f *majorizes* g, denoted by $f \geq g$, if $\mathrm{dom}(g) \subseteq \mathrm{dom}(f)$ and $f(x) \geq g(x)$ for all $x \in \mathrm{dom}(g)$. A function f is *(weakly) increasing* (or *monotone*) if, for every pair of elements $x, y \in \mathrm{dom}(f)$, $x < y$ implies $f(x) \leq f(y)$, and a *strictly increasing* function f is obtained simply by replacing the above condition $f(x) \leq f(y)$ with $f(x) < f(y)$. Similarly, we can define *(weakly) decreasing* functions and *strictly decreasing* functions. A function f is *unbounded* if, for every x, there exists an element $y > x$ such that $f(y) > f(x)$. A function f on Σ^* is called *length-increasing* if $|f(x)| > |x|$ for all $x \in \Sigma^*$, and f is *length-preserving* if $|f(x)| = |x|$ for all $x \in \Sigma^*$.

A function f from $\mathrm{dom}(f)$ to $\mathbb{R}$ is *convex* if, for any $x, y \in \mathrm{dom}(f)$ and any real number $\gamma \in [0, 1]$,

$$f(\gamma \cdot x + (1 - \gamma) \cdot y) \leq \gamma \cdot f(x) + (1 - \gamma) \cdot f(y),$$

and f is *concave* if we replace the symbol $\leq$ by $\geq$ in the above inequality.

A function f on Σ^* is *polynomially honest* (p-honest, for short) if there is a polynomial p such that $|x| \leq p(|f(x)|)$ for all x. Similarly, a function f on Σ^* is *exponentially honest* (exp-honest, for short) if there is a constant $c > 0$ such that $|x| \leq 2^{c|f(x)|+c}$ for all x.

Traditionally, a function f on Σ^* is called *polynomially bounded* (p-bounded, for short) if there exists a polynomial p such that $|f(x)| \leq p(|x|)$ for all strings x. A function f from Σ^* to $\mathbb{R}^+$ is called *polynomially bounded* (p-bounded, for short) if there exists a polynomial p such that $f(x) \leq p(|x|)$ for all x [36]. Note that any composition of two p-bounded functions is also p-bounded. Similarly, *exponentially bounded* (exp-bounded, for short) functions are defined by replacing $p(n)$ as above by an exponential $2^{p(n)}$.

A function f from $\mathrm{dom}(f)$ to $\mathbb{R}$ is *positive* if $f(x) > 0$ for all $x \in \mathrm{dom}(f)$. Given a subset S of $\mathrm{dom}(f)$, we say that f is *positive on* S if $f(x) > 0$ for all $x \in S$.

For any functions f and g mapping to $\mathbb{R}^+$, we denote by $f \times g$, $f + g$, $\min\{f, g\}$, and $\max\{f, g\}$ the functions defined, respectively, as follows: for all x, $(f \times g)(x) = f(x) \cdot g(x)$, $(f + g)(x) = f(x) + g(x)$, $\min\{f, g\}(x) = \min\{f(x), g(x)\}$, and $\max\{f, g\}(x) = \max\{f(x), g(x)\}$.

For a function f from $\mathbb{N}$ to $\mathbb{R}^+$, f is *negligible* if, for every positive polynomial p, it holds that $f(n) < \frac{1}{p(n)}$ for almost all natural numbers n.

For two integers a and b, the notation $a|b$ means that there exists an integer c satisfying $b = c \cdot a$. The equivalence relation of *congruence* modulo n, is defined as follows: two integers a and b are *congruent modulo* n if $n|(a - b)$, and this is denoted by $a \equiv b \pmod{n}$.

Let f be a function from $\mathbb{N}$ (or $\mathbb{R}$) to $\mathbb{R}$, and let $r \in \mathbb{R}$. If, for every real number $\epsilon > 0$, there exists a number

$x_0 \in \mathbb{N}$ (or $x_0 \in \mathbb{R}$) such that $|f(y) - r| < \epsilon$ for all y in $\mathbb{N}$ (or $\mathbb{R}$) larger than x_0, we write $\lim_{x\to\infty} f(x) = r$. Analogously, for an increasing function f from Σ^* to $\mathbb{R}^+$, the notation "$\lim_{x\to\infty} f(x) = r$" means that

(i) $f(x) \leq r$ for all $x \in \Sigma^*$; and

(ii) for every real number s with $s < r$, there exists a string x such that $s \leq f(x)$.

2.2.6 Asymptotic Notation

We often use $O(\cdot)$ (big oh), $o(\cdot)$ (little oh), $\Omega(\cdot)$ (big omega), $\omega(\cdot)$ (little omega), and $\Theta(\cdot)$ (theta) as sets of functions. Let f be a function from $\mathbb{N}$ to $\mathbb{R}^+$. We formally define five sets, $O(f)$, $o(f)$, $\Omega(f)$, $\omega(f)$, and $\Theta(f)$:

1. $O(f)$ is the set of functions h such that, for *some* constant $c > 0$, $h(n) \leq c \cdot f(n)$ for almost all n.
2. $o(f)$ is the set of functions h such that, for *every* constant $c > 0$, $h(n) \leq c \cdot f(n)$ for almost all n.
3. $\Omega(f)$ is the set of functions h such that, for *some* constant $c > 0$, $c \cdot f(n) \leq h(n)$ for almost all n.
4. $\omega(f)$ is the set of functions h such that, for *every* constant $c > 0$, $c \cdot f(n) \leq h(n)$ for almost all n.
5. $\Theta(f) = O(f) \cap \Omega(f)$.

To emphasize the variable n used for the function f, we also write $O(f(n))$ for $O(f)$ and similarly for the other four sets.

For example, $\lambda n.2n \in o(n^2)$ but $\lambda n.2n^2 \notin o(n^2)$; $\lambda n.n^2/2 \in \omega(n)$ but $\lambda n.n^2/2 \notin \omega(n^2)$.

Definition 2.2.1 We define the following three notations:

1. $n^{O(1)} = \bigcup_{k>0} O(n^k)$.
2. $2^{O(n)} = \bigcup_{k>0} O(2^{kn})$.
3. $2^{n^{O(1)}} = \bigcup_{k>0} O(2^{n^k})$.

Traditionally, the notations $O(f(n))$, etc. , are defined as "pseudo"-functions: the notation "$g(n) = O(f(n))$," for example, means that g is in $O(f(n))$. In this thesis, we follow this convention and loosely use the notations $O(\cdot)$, etc. , as if they are "functions." As an example, when we write that $n! = O(\sqrt{2\pi n}(\frac{n}{e})^n)$, we actually mean that the function $\lambda n.n!$ belongs to $O(\sqrt{2\pi n}(\frac{n}{e})^n)$.

2.2.7 Probability Measure

We begin with the formal definitions of probability theory.

A *sample space* Ω is an underlying set. This thesis uses a subset of Σ^∞ as a sample space Ω. A *σ-field* $(\Omega, \mathbb{F})$ consists of a sample space Ω and a subset $\mathbb{F}$ of $\mathcal{P}(\Omega)$ satisfying the following conditions:

(i) $\emptyset \in \mathbb{F}$;

(ii) $\mathcal{E} \in \mathbb{F}$ implies $\overline{\mathcal{E}} \in \mathbb{F}$, where $\overline{\mathcal{E}} = \Omega - \mathcal{E}$; and

(iii) $\{\mathcal{E}_i\}_{i \in \mathbb{N}} \subseteq \mathbb{F}$ implies $\bigcup_{i \in \mathbb{N}} \mathcal{E}_i \in \mathbb{F}$.

Any set in $\mathbb{F}$ is referred to as an *event*.

A *probability measure* $\mathbf{Pr}$ is a function from $\mathbb{F}$ to $[0,1]$ that satisfies the following conditions:

(i) for all set $A \subseteq \Omega$, $0 \leq \mathbf{Pr}[A] \leq 1$;

(ii) $\mathbf{Pr}[\Omega] = 1$; and

(iii) (Countable Additivity) for mutually disjoint events $\{\mathcal{E}_i\}_{i \in \mathbb{N}}$, $\mathbf{Pr}[\bigcup_{i \in \mathbb{N}} \mathcal{E}_i] = \sum_{i \in \mathbb{N}} \mathbf{Pr}[\mathcal{E}_i]$.

For an event $\mathcal{E}$, the notation $\mathbf{Pr}[\mathcal{E}]$ denotes the *probability* of $\mathcal{E}$. A *support* of $\mathbf{Pr}$ is any $\mathbb{F}$-set A for which $\mathbf{Pr}[A] = 1$.

The *conditional probability* of $\mathcal{E}_1$ given $\mathcal{E}_2$ is denoted by $\mathbf{Pr}[\mathcal{E}_1 \mid \mathcal{E}_2]$ and is given by:

$$\frac{\mathbf{Pr}[\mathcal{E}_1 \cap \mathcal{E}_2]}{\mathbf{Pr}[\mathcal{E}_2]}$$

assuming that $\mathbf{Pr}[\mathcal{E}_2] > 0$.

A *probability space* is a triplet $(\Omega, \mathbb{F}, \mathbf{Pr})$, where $(\Omega, \mathbb{F})$ is a σ-field and $\mathbf{Pr}$ is a probability measure defined on the sample space Ω. When Ω is clear from the context, Ω may be omitted.

A collection of events $\{\mathcal{E}_i\}_{i \in I}$, where I is an index set, is *independent* if, for all subsets $S \subseteq I$, $\mathbf{Pr}[\bigcap_{i \in S} \mathcal{E}_i] = \prod_{i \in S} \mathbf{Pr}[\mathcal{E}_i]$; or equivalently, $\mathbf{Pr}[\mathcal{E}_j \mid \bigcap_{i \in S} \mathcal{E}_i] = \mathbf{Pr}[\mathcal{E}_j]$ for all $j \in I$. Similarly, $\{\mathcal{E}_i\}_{i \in I}$ is *pairwise independent* if, for any pair $\{i, j\} \subseteq I$, $\mathbf{Pr}[\mathcal{E}_i \cap \mathcal{E}_j] = \mathbf{Pr}[\mathcal{E}_i] \cdot \mathbf{Pr}[\mathcal{E}_j]$.

A *(discrete) random variable* X is a function over the sample space Ω whose range D is either a finite or countable infinite subset of $\mathbb{R}$ such that, for all $x \in D$, $\{w \in \Omega \mid X(w) \leq x\} \in \mathbb{F}$. By identifying Σ^* with $\mathbb{N}$, we can introduce discrete random variables whose ranges are particular subsets of Σ^*.

The *expected value* or *expectation* of a random variable X is denoted by $E[X]$ and is defined by $\sum_{x \in \Omega} x \cdot \mathbf{Pr}[X = x]$.

In this thesis, we deal mainly with *discrete probability measure* on a σ-field with a sample space $\Omega \subseteq \Sigma^\infty$, and the notation $\mathbf{Pr}[\cdot]$ will be reserved to denote the "uniform" probability measure. For more details on probability theory, the reader may refer to a text devoted to the subject, for example [11].

2.3 Models of Computation

As a model of "computation," we focus on *Turing machines* which were introduced by A. Turing and E. Post in the 1930's. This thesis uses the standard models of Turing machines with a finite number of *semi-infinite* tapes (i.e., the tape has a leftmost square but is infinite to the right).

We shall informally use the terms "algorithms" and "algorithmically computable" in this thesis. Although there is no precise definition for these terms, we stand on the common belief, known as Church's Thesis, that algorithms are described by Turing machines.

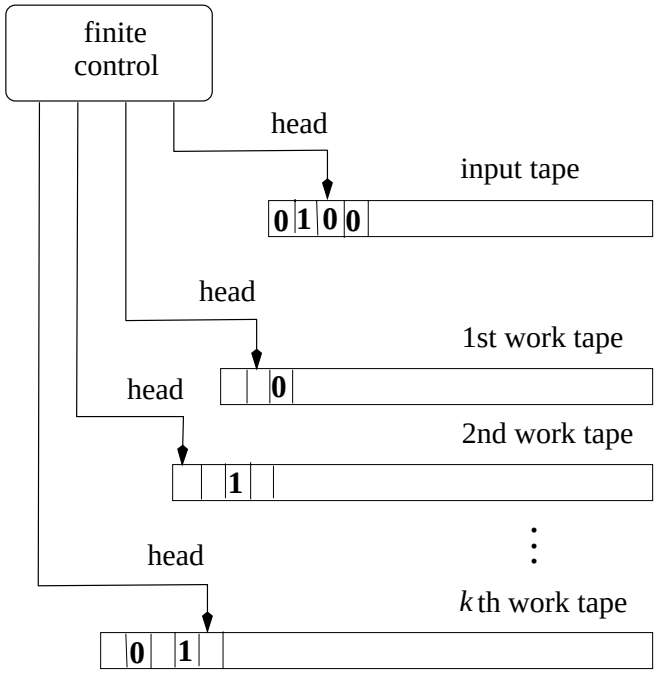

Figure 2.1: The hardware of a Turing machine

2.3.1 Deterministic Turing Machines

A k-tape off-line *deterministic Turing machine* is formally a sextuple $\langle Q, \Sigma, q_0, ACC, REJ, \delta \rangle$ which consists of Q, a finite set of states; Σ, a tape alphabet with a special symbol for the blank; q_0, an initial state; ACC, a set of accepting states; REJ, a set of rejecting states; and δ, a transition function from $Q \times \Sigma^k$ to $\Sigma^{k-1} \times Q \cup ACC \cup REJ \times \{R, N, L\}^k$. Figure 2.1 illustrates the hardware of a Turing machine.

Turing machines are casually called just *machines*. The function δ is considered a *program* (or an *algorithm*) for the machine, and we often identify a Turing machine with its program (or algorithm).

A *configuration* (or *instantaneous description*) of a machine M is a description which contains the contents of each tape, the position of each tape head, and the state of the machine. The *initial configuration* is a configuration in which the input tape contains an input, other tapes are blank, the internal state of the machine is the initial state, and all head positions on tapes are the leftmost squares. An *accepting (rejecting, resp.) configuration* of M on x is a configuration of M on x whose state is an accepting (rejecting, resp.) state. A *halting configuration* of M is either an accepting or a rejecting configuration, i.e., a configuration from which no other configurations can be reached by the transition function.

To describe how the machine works, we need a concept of "computation."

Definition 2.3.1 (Computation) A *computation* of M on input x is a (finite or infinite) sequence of configurations of M such that:

(i) it starts with the initial configuration of M on x;

(ii) each step from a configuration to another configuration is made by the transition function; and

(iii) if finite, it ends in a halting configuration of M on x.

An *accepting (rejecting,* resp.*) computation* is a computation which terminates in an accepting (rejecting, resp.) configuration.

A deterministic Turing machine M *accepts* an input x if there is an accepting computation of M on input x; otherwise, M *rejects* x. Denote by $L(M)$ the set of all strings which are accepted by M.

Let $M(x)$ denote the output of a machine M on input x if it exists. The *running time* of M on input x is the length of the computation of M on x, and we denote by $\mathrm{Time}_M(x)$ the running time of M on x. In the case where the computation is not finite, we set $\mathrm{Time}_M(x) = \infty$.

Definition 2.3.2 (Time/Space Constructible) A function f on $\mathbb{N}$ is called *time-constructible* if there exists a deterministic Turing machine M which, on input 1^n, terminates after exactly $f(n)$ steps are made. A function f on $\mathbb{N}$ is *space-constructibility* if there exists a deterministic Turing machine which, on input 1^n, it marks the $f(n)$th square of the first work tape (among a finite number of work tapes).

2.3.2 Nondeterministic Turing machines

Another important model of computation is "nondeterministic" Turing machines. A *nondeterministic Turing machine* is a variant of deterministic Turing machines with the exception that the transition function δ is a map from $Q \times \Sigma^k$ to $\mathcal{P}(\Sigma^{k-1} \times Q \cup ACC \cup REJ \times \{R, N, L\}^k)$.

As for nondeterministic Turing machines, we alter the definition of "computation" to a set of "computations," a so-called "computation tree." A *computation tree* of M on input x is a tree whose nodes are configurations of M on x, in which the root of the tree is the initial configuration, and the children of each node are such configurations that are reachable from the node in one step by the transition function. All configurations following each configuration by a single application of the transition function is called *nondeterministic choices* if the number of such configurations is more than 1. An *accepting (rejecting,* resp.*) computation* is a path from the root to a leaf which ends with an accepting (rejecting, resp.) configuration.

The accepting criteria of nondeterministic Turing machines is similar to that of deterministic machines and is determined by the existence of an accepting computation. More precisely, the machine M *accepts* x if there exists an accepting computation of M on input x; otherwise, the machine *rejects* x.

A nondeterministic Turing machine for which the number of accepting paths on each input is at most one is called *unambiguous* [103].

In general, average-case complexity measure is sensitive to the definition of time-complexity of nondeterministic Turing machines (see [97]), and we should pay careful attention to the definition of the running time of the machine when dealing with nondeterministic computations.

In worst-case complexity theory, the running time of a nondeterministic Turing machine is often defined to be the minimal length of all accepting computation paths if one exists; otherwise, it is defined to be

1. For time/space constructible complexity bounds (such as "polynomial-time" or "logarithmic-time"), we may assume that all computation paths on each input are of the same length, and by convention let the running time be the maximal length of any computation path, since time-constructible complexity bounds t guarantee that we can modify any machine having this bound to a machine for which the length of any computation path is exactly t. This is explained as follows: every time-bounded Turing machine is designed in such a way that each has an internal clock (which does not access oracles), and this clock adjusts the running time of the machine no matter what computation path it follows. We call the machines equipped with internal clocks *clocked Turing machines*.

Summing up, there are three models of nondeterministic Turing machines together with their running-time criteria:

(i) A model of nondeterministic Turing machines with traditional measurement of running time, namely, the shortest accepting path if one exists, or else 1 (or equivalently, taking the shortest rejecting path);

(ii) A model of nondeterministic Turing machines with strict measurement of running time, that is, the shortest accepting path if one exists, or else, the longest rejecting path;

(iii) A model of *clocked* nondeterministic Turing machines.

As long as the running time of a machine on an accepting computation path is bounded above by some *time-constructible function* (most time-bounded complexity classes in worst-case complexity theory satisfy this condition), all these definitions are essentially equivalent (neglecting constant slowdown). Since average-case complexity theory does not require this condition, the choice of a model is very important and often leads us to different consequences. In later chapters, we shall discuss the choice of models and possible consequences.

Historically, Goldreich [30] first discussed the average running time of nondeterministic Turing machines and used a model of nondeterministic Turing machines, the lengths of whose computation paths are measured by some time-bounded deterministic Turing machines. His definition is actually equivalent to choosing model (iii) as described above. Later Wang and Belanger [111], and also Schuler and Yamakami [97] presented interesting results based on model (iii). In particular, Schuler and Yamakami [97] constructed an average-case version of the (worst-case) polynomial-time hierarchy based on model (i), but the average-case hierarchy obtained here does not seem to be a proper analogue of the worst-case hierarchy (it lacks some properties like $\mathbf{NP}^{\mathbf{P}} = \mathbf{NP}$). In this thesis, we choose the most general model (i), even though the model does not seem to provide the property that time-bounded nondeterministic computations can be simulated by space-bounded deterministic machines of the same complexity.

Definition 2.3.3 (Running Time of Nondeterministic Turing Machines) For a nondeterministic Turing machine M, the *running time* of M on input x, $\mathrm{Time}_M(x)$, is defined to be the length of the shortest accepting computation path of M on x if one exists; otherwise, it is defined to be 1.

2.3.3 Oracle Turing Machines

To speed up the computation of an algorithm or to make it as accurate as possible, we need a supplementary source of information which the algorithm can retrieve and use. Such a source is called an *oracle*, and a Turing machine equipped with a system retrieving information from an oracle is called an *oracle Turing machine*. An oracle Turing machine makes a *query* to an oracle and receives its answer in a single step. We start by describing those notions formally.

An *oracle Turing machine* is a Turing machine with the following additional devices: a distinguished tape, a so-called *oracle tape* or *query tape*, and three distinguished states , QUERY, YES, and NO. A *computation (tree)* of an oracle machine M with an oracle (set) on input x is defined in a way similar way to that for "non-oracle" Turing machine except that it incorporates oracle queries. Initially the query tape is blank. If the machine M enters the QUERY state, then in a single step, M queries a string to the oracle which appears on the query tape; if this string belongs to the oracle set, then M enters the YES state; otherwise, M enters the NO state. Immediately after each oracle query, the query tape becomes blank.

We can easily extend the definition of oracle Turing machines with *set oracles* (or oracle sets) into those equipped with *function oracles* f. The oracle machine has the QUERY state and the YES state; if it makes a query z to an oracle, then the oracle returns the value $f(z)$ of the function f in a single step and the machine enters the YES state; at the same time, the head of the query tape is moved to the leftmost square of the tape.

Since oracle Turing machines with the *empty oracle* (i.e., the empty set) can be easily translated into non-oracle Turing machines (because we know the oracle answers), we often identify such oracle machines with non-oracle ones. In this sense, without loss of generality, we can view non-oracle Turing machines as a special case of oracle Turing machines. Therefore, subsequent definitions will be stated only for oracle machines without repeating similar definitions for non-oracle machines.

Definition 2.3.4 (Adaptive/Nonadaptive Query) An oracle Turing machine M is said to make *nonadaptive queries* if, on each computation path, M produces a list (called a *query list*) of all strings which are possibly queried before the first query is made. Otherwise, M is said to make *adaptive queries*.

A query list provides us with sufficient information about which strings will possibly be queried in future computations. We remark that it is not necessary for an oracle machine to query all the strings in the query list.

Let $\mathrm{Acc}(M, A, x)$ denote the set of (codes of) accepting computation paths of M on input x with oracle A, and similarly $\mathrm{Rej}(M, A, x)$ denotes that of rejecting computation paths. Let $Q(M, A, x, y)$ be the set of strings queried by M with oracle A on input x on computation path y. If M is deterministic, then we simply denote by $Q(M, A, x)$ the set of all strings queried by M on input x with oracle A.

By $L(M, A)$ we denote the set of strings accepted by M with oracle A, and we simply say that M with oracle A *recognizes* (or *accepts*) a set B if $B = L(M, A)$. For a machine M, $M^A(x)$ denotes the output of a computation of M on input x. For a deterministic Turing machine M with an output tape (also called a

transducer), we say that M *computes* a function f if $f(x) = M^A(x)$ for all $x \in \Sigma^*$.

By $\mathrm{Time}_M^A(x)$, we denote the running time of machine M with oracle A on input x. Similarly, $\mathrm{Space}_M^A(x)$ denotes the tape space used by M with oracle A on input x. Technically speaking, there are two possible definitions for $\mathrm{Space}_M^A(x)$ depending on whether the space of the query tape is counted. This possibly changes the power of relativized space-bounded complexity classes, such as **PSPACE**. In this thesis, we take all tapes including query tapes into consideration in order to measure the tape space used by the machine M with oracle A.

2.3.4 Alternating Turing Machines

The notion of *alternating Turing machines* was introduced by Chandra, Kozen, and Stockmeyer [21] as an extension of nondeterministic Turing machines.

Each machine is equipped with extra states, called $\forall$ (universal) and $\exists$ (existential). Each configuration in a finite tree of computation for an alternating Turing machine is labeled as either *universal* ($\forall$) or *existential* ($\exists$), according to the states of the machine. Next we define an accepting computation tree. First we recursively determine the *yes-configurations*:

(i) a halting configuration is a *yes*-configuration if it is an accepting configuration;

(ii) a non-halting $\exists$-configuration is *yes*-configuration if at least one of its children is so; and

(iii) a non-halting $\forall$-configuration is a *yes*-configuration if all of its children are so.

For convenience, configurations which are not *yes*-configurations are called *no-configurations*. An *accepting computation tree* T' of M on input x is a subtree of a computation tree T of M on x satisfying the following conditions:

(i) all configurations in T' are *yes*-configurations;

(ii) an existential configuration in T' has one child node in T; and

(iii) a universal configuration in T' has all of its children in T.

The machine *accepts* an input if there exists an accepting computation tree (equivalently, the root of the computation tree has a *yes*-configuration); otherwise, the machine *rejects* the input.

Alternation is the maximum over, all computation paths from the root to a leaf, of the number of times in which different labels of configurations (i.e., $\exists$- or $\forall$-configurations) change. Note that, by convention, the initial configuration is assumed to contribute the first alternation. For example, nondeterministic Turing machines have 1-alternation because all configurations are $\exists$-configurations.

Here we shall introduce a new variant of alternating Turing machines, so-called "semi-deterministic" alternating Turing machines, which embodies deterministic computations relative to some alternating Turing machines. This notion is useful for describing the Δ-level of the polynomial-time hierarchy, for example (see Section 2.5). A *semi-deterministic alternating Turing machine* is, roughly speaking, an alternating Turing

machine which puts an additional restriction on its computation tree. For simplicity, we assume that the number of nondeterministic choices is 2. Then, any computation path can be encoded as a binary string. We label each computation path with this code to distinguish individual computation paths. For brevity, write s_j^m to denote the jth string of the set Σ^m in the standard order (N.B. 0^m is the 0th string of Σ^m).

Definition 2.3.5 (Semi-Deterministic Alternating Turing Machines) A *semi-deterministic Turing machine* is an alternating Turing machine with the following constraints for the first three series of $\forall$- and $\exists$-configurations of the computation trees T_x of the machine on each input x.

On input x, the machine starts with an $\exists$-state, and during this state the machine produces computation subpaths. Let p_1 be any such computation subpath and suppose that the machine makes k queries. By the above coding scheme, the subpath p_1 is labeled by $b = b_1 b_2 \cdots b_k$, where $b_i \in \{0,1\}$, $1 \leq i \leq k$. Next the machine enters an $\forall$-state in which it produces exactly 2^m branches for some $m \geq \log k$, each of which is labeled by an m bit string $c = c_1 c_2 \cdots c_m$, where $c_i \in \{0,1\}$, $1 \leq i \leq m$. Let p_2 be any such a branch following p_1. Consider the current configuration, called *critical*, and denote it by d_l, where l is the label bc attached to the subpath $p_1 p_2$. At the next step, the machine chooses either an $\exists$-state or $\forall$-state, depending on the label l $(= b_1 \cdots b_k c_1 \cdots c_m)$. The machine enters an $\exists$-state if $b_j = 1$ and $c = s_j^m$ for some j; otherwise, it enters an $\forall$-state.

All critical configurations of the computation tree T_x satisfy the following two conditions on their *labels* with respect to yes- or no-configurations: for any two critical configurations d_l and $d_{l'}$, where $l = b_1 \cdots b_k c_1 \cdots c_m$ and $l' = b'_1 \cdots b'_{k'} c'_1 \cdots c'_{m'}$, assuming that $c = s_j^m$ and $c' = s_j^{m'}$ for some j with $0 \leq j < \min\{k, k'\}$,

(i) if $b_1 \cdots b_{j+1} = b'_1 \cdots b'_{j+1}$, then configurations d_l and $d_{l'}$ have the same label: and

(ii) if $b_1 \cdots b_j = b'_1 \cdots b'_j$ but $b_1 \cdots b_{j+1} \neq b'_1 \cdots b'_{j+1}$ then configurations d_l and $d_{l'}$ have different labels.

A semi-deterministic Turing machine M is said to have *k-alternations* if the maximal alternation of all computation trees of M is at most $k+2$ (because the first two alternations are fixed and should not be counted).

We define the running time of an alternating Turing machine as follows.

Definition 2.3.6 (Running Time of Alternating Turing Machines) The running time of an alternating Turing machine on input x is the minimal height of accepting computation trees of M on x if M accepts x; otherwise, the running time is defined to be 1.

2.3.5 Worst-Case Time/Space Complexity

In worst-case complexity theory, the time (or space) complexity of an algorithm is often considered as a function defined on the natural numbers $\mathbb{N}$ because we are interested only in the instances of each length which are hard to compute.

Definition 2.3.7 Let M be an oracle Turing machine and A a set. Let t be a function on $\mathbb{N}$, and let $\mathcal{T}$ be a set of functions on $\mathbb{N}$.

1. The Turing machine M with oracle A is called *t-time bounded* (or a *t-time Turing machine*, for short) if $\mathrm{Time}_M^A(x) \leq t(|x|)$ for all x. Similarly, M with oracle A is *t-space bounded* (or a *t-space Turing machine*) if $\mathrm{Space}_M^A(x) \leq t(|x|)$ for all x.

2. The oracle machine M is *$\mathcal{T}$-time bounded* (or a *$\mathcal{T}$-time Turing machine*) if M with oracle A is t-time bounded for some t in $\mathcal{T}$. The notion of a *$\mathcal{T}$-space Turing machine* is defined analogously.

Here we use conventional abbreviations: if $\mathcal{T}$ is the set of polynomials, we say that M is *polynomial-time (*or *polynomial-space) bounded*, and similarly, if $\mathcal{T}$ is the set of exponentials (linear-exponentials, logarithms, resp.), M is called *exponential-time* (*linear-exponential-time*, *logarithmic-space*, resp.) *bounded*.

2.4 Randomized Algorithms

In this thesis, we would like to use two different terms, *randomized Turing machines* and *probabilistic Turing machines*, to cope with randomized algorithms.

A *randomized Turing machine* is a model of randomized computation and a variant of a nondeterministic Turing machine with no accepting criteria. Intuitively, we equip a machine with a special mechanism for generating an unbiased coin flip in one step which determines the choice of the next configurations. More precisely, a randomized Turing machine is a Turing machine with a distinguished state, called *coin-tossing state*, in which the finite control unit specifies two possible next states. The (finite or infinite) *computation* of a randomized Turing machine is determined by its input as well as by the outcomes of the coin tosses performed by the machine.

We can view randomized Turing machines as *partial* functions with two variables, one of which is a "usual" input initially written on the input tape, and the other of which is a *random input* (or *random seed*), that is a binary sequence representing the outcomes of the coin tosses. Following Blass and Gurevich [12], we shall formalize this model below.

A randomized Turing machine M is equipped with an auxiliary semi-infinite read-only tape, called a *random tape* which may consist of an *infinite* sequence r of "random" bits (i.e., $r \in \Sigma^\infty$). The head on the random tape can move only to the right and cannot stay at the same square after the machine reads a symbol on the tape. The machine's access to the random tape corresponds to a coin-flip, and we may say that the machine *flips a (fair) coin* when it accesses the random tape. For each infinite random sequence r, let $Read_{M,x}(r)$ be the initial segment of r that is read by M on input x during its computation. Note that if a computation is finite, then $Read_{M,x}(r)$ is also finite, but not conversely. Let the sample space $\Omega_M(x)$ be

$$\Omega_M(x) = \{Read_{M,x}(r) \mid r \in \Sigma^\infty\}.$$

Recall that $\mathbf{m}$ is the Lebesgue measure on the line $\mathbb{R}$. We define the probability measure $\mathbf{Pr}$ on the sample

space $\Omega_M(x)$ as follows: for any event $E \subseteq \Omega_M(x)$,

$$\mathbf{Pr}[E] = \mathbf{m}(\{r \in \Sigma^\infty \mid Read_{M,x}(r) \in E\})$$

by identifying $[0,1]$ with Σ^∞. Throughout this thesis, we use $\mathbf{Pr}[\cdot]$ to denote this probability measure. In particular, if event E consists only of finite strings, then

$$\mathbf{Pr}[E] = \sum_{s \in E} 2^{-|s|}.$$

Abusing notation, we let Ω_M denote the set $\{(x,s) \mid s \in \Omega_M(x)\}$. (Note that Ω_M is *not* a sample space.)

We shall define the running time of the machine M on input x as follows. For any input $(x,r) \in \Sigma^* \times \Sigma^\infty$, let $T_M(x,r)$ denote the time taken by M on input x and random input r. Whenever M does not halts, let $T_M(x,r) = \infty$. Next let $\mathrm{Time}_M(x;\cdot)$ to be the (partial) function from $\Omega_M(x)$ to $\mathbb{N}$ such that $T_M(x,r) = \mathrm{Time}_M(x; Read_{M,x}(r))$ for all $r \in \Sigma^\infty$. For convenience, if $(x,r') \notin \Omega_M$, then $\mathrm{Time}_M(x;r')$ is undefined. Unless stated otherwise, the notation $\lambda xs.\mathrm{Time}_M(x;s)$ is used to mean the *total* function defined as above from Γ_M to $\mathbb{N} \cup_\infty$.

For each input x to a randomized Turing machine, let us denote by $M(x)$ a random variable over the sample space $\Omega_M(x)$. Let Q be a property on $\{0,1\}$. Then we denote by $\mathbf{Pr}_M[Q(M(x))]$ the probability that $Q(M(x))$ holds. For the sake of convenience, we also use the notation $M(x;r)$ to mean the output of a computation by M on input x with random seed r when r is in $\Omega_M(x)$. In the case where $r \notin \Omega_M(x)$, let $M(x;r)$ be undefined. Thus, $M(x;r)$ is a partial function. The notation $\mathbf{Pr}_s[Q(M(x;s))]$ and $\mathbf{Pr}_s[Q(M(x;s)) \mid s \in \Omega_M(x)]$ are also used to denote $\mathbf{Pr}_M[Q(M(x))]$.

2.4.1 Random-Input Domains

Blass and Gurevich [12] created a general framework for the average analysis of randomized algorithms based on the notion of "dilations." Here we take a simplified approach.

Definition 2.4.1 (Random-Input Domain) A subset Γ of $\Sigma^* \times \Sigma^*$ is called a *random-input domain* if, for all $x, s, s' \in \Sigma^*$, $(x,s) \in \Gamma$, $s' \sqsubseteq s$, and $s \neq s'$ imply $(x,s') \notin \Gamma$. For each string x, we set $\Gamma(x) = \{s \mid (x,s) \in \Gamma\}$.

It is important to note that $\Gamma(x)$ consists only of *finite* strings and that it may not be a sample space with respect to $\mathbf{Pr}$ because $\mathbf{Pr}[\Gamma(x)] \leq 1$.

Definition 2.4.2 (Rarity Functions) [12] Let Γ be a random-input domain. The *rarity function* of Γ is denoted by U_Γ and defined by

$$U_\Gamma(x) = \frac{1}{\sum_{s \in \Gamma(x)} 2^{-|s|}}$$

for all x. If the rarity function U_Γ satisfies $U_\Gamma(x) = 1$ for all x, then we call Γ *almost total*.

In other words, $1/U_\Gamma(x) = \mathbf{Pr}[\Gamma(x)]$.

For a randomized Turing machine M, let $\Gamma_M = \{(x,s) \in \Omega_M \mid s \text{ is finite}\}$. In particular, when Γ_M is almost total, there is no need to differentiate between $\Gamma_M(x)$ and the sample space $\Omega_M(x)$ since $\mathbf{Pr}[\Gamma_M(x)] = \mathbf{Pr}[\Omega_M(x)] = 1$.

Definition 2.4.3 (Random Functions) A *random function* f is a function from a random-input domain to $\mathbb{R}^{+\infty}$. A random function f with its random-input domain Γ is called *almost total* if Γ is almost total and $\mathbf{Pr}_s[f(x,s) < \infty \mid s \in \Omega_M(x)] = 1$, where the subscript s in this equation emphasizes the random variable over $\Omega_M(x)$. For a random function f with its random-input domain Γ, the *(conditional) expectation* of f on input x, symbolically $E_s[f(x,s) \mid s \in \Gamma(x)]$, is defined by

$$\sum_{s \in \Gamma(x)} U_\Gamma(x) \cdot f(x,s) \cdot 2^{-|s|}.$$

For example, the partial function $\lambda xs.\mathrm{Time}_M(x;s)$ for a randomized Turing machine M is a random function from Ω_M to $\mathbb{N}$.

2.4.2 Probabilistic Turing Machines

The notion of probabilistic Turing machines was proposed by de Leeuw, Moore, Shannon, and Shapiro [23] in 1955. Pioneer works on probabilistic Turing machines were done by Gill [28] and Santos [86].

A *probabilistic Turing machine* M is a randomized Turing machine with the following accepting criteria: for every x, either $\mathbf{Pr}_s[M(x;s) = 1 \mid s \in \Omega_M(x)] > \frac{1}{2}$ or $\mathbf{Pr}_s[M(x;s) = 0 \mid s \in \Omega_M(x)] > \frac{1}{2}$ [28]. We say that M *accepts* x if $\mathbf{Pr}_s[M(x;s) = 1 \mid s \in \Omega_M(x)] > \frac{1}{2}$, and M *rejects* x if $\mathbf{Pr}_s[M(x;s) = 0 \mid s \in \Omega_M(x)] > \frac{1}{2}$. By this definition, the rarity function U_{Γ_M} of Γ_M is bounded above by 2 because

$$\frac{1}{U_{\Gamma_M}(x)} = \sum_{s \in \Gamma_M(x)} 2^{-|s|} = \mathbf{Pr}[\Gamma_M(x)] \geq \mathbf{Pr}_s[\, M(x;s) \text{ halts } \mid s \in \Omega_M(x)] > \frac{1}{2}.$$

For a probabilistic Turing machine M, let $L(M)$ denote the set of all strings which are accepted by M. We say that M *recognizes* D if $D = L(M)$.

Let D be the set recognizable by a probabilistic Turing machine M. The *error probability* of M for D is the function e_M defined by

$$e_M(x) = \mathbf{Pr}_M[M(x) \neq \chi_D(x)],$$

where χ_D is the characteristic function for D. Clearly $e_M(x) < \frac{1}{2}$. We say that M recognizes D with *bounded error probability* [28] if there is a constant ϵ with $0 \leq \epsilon < \frac{1}{2}$ such that $e_M(x) \leq \epsilon$ for every x. We simply call M a *bounded-error probabilistic Turing machine* if M recognizes some set with bounded-error probability.

A probabilistic Turing machine M is said to make a *one-sided error* if it satisfies the additional condition: if M rejects x, then all of its finite computation paths terminate in rejecting configurations. Such a machine is sometimes referred to as a *random Turing machine*. In contrast, the previous probabilistic Turing machines are said to make *two-sided errors*.

Another type of measure was given by Gill [28]. For a probabilistic Turing machine M, let D be the set accepted by M. Set

$$\mathrm{Time}^*_M(x) = \begin{cases} \min\{n \mid \mathbf{Pr}_s[M(x;s) = \chi_D(x) \text{ within } n \text{ steps } \mid s \in \Gamma_M(x)] > \frac{1}{2}\} & \text{if one exists,} \\ \infty & \text{otherwise.} \end{cases}$$

Note that the term $\Gamma_M(x)$ can be replaced by $\Omega_M(x)$ because $\mathbf{Pr}_M[M(x) = \chi_D(x)] \geq 1/2$.

A relationship between two measures $\lambda xs.\mathrm{Time}_M(x;s)$ and $\lambda x.\mathrm{Time}^*(x)$ is best described in the following lemma.

Lemma 2.4.4 **[28]** *Let δ be a real number satisfying $0 < \delta \leq 1$. For every bounded-error probabilistic Turing machine M, there exists a constant $c > 0$ such that $\mathrm{Time}^*_M(x) \leq c \cdot E_s[\mathrm{Time}_M(x;s)^\delta \mid s \in \Gamma_M(x)]^{1/\delta}$ for all strings x. In particular, $\mathrm{Time}^*_M(x) \leq c \cdot E_s[\mathrm{Time}_M(x;s) \mid s \in \Gamma_M(x)]$.*

Proof. Let ϵ be the error probability bound of M. Hence, $0 \leq \epsilon < \frac{1}{2}$. Let $c = \frac{2}{1-2\epsilon}$ and $c' = c^{1/\delta}$. For simplicity, write $h(x) = E_s[\mathrm{Time}_M(x;s)^\delta \mid s \in \Gamma_M(x)]$.

Let x be an arbitrary string. By Lemma A.1,

$$\mathbf{Pr}_s[\mathrm{Time}_M(x;s)^\delta > c \cdot h(x) \mid s \in \Gamma_M(x)] < \frac{1}{c}.$$

This is equivalent to

$$\mathbf{Pr}_s[\mathrm{Time}_M(x;s) > c' \cdot h(x)^{1/\delta} \mid s \in \Gamma_M(x)] < \frac{1}{c}.$$

Let D be the set accepted by M.

$$\mathbf{Pr}_s[M(x;s) \neq D(x) \text{ in time } c' \cdot h(x)^{1/\delta} \mid s \in \Gamma_M(x)] \;\leq\; \mathbf{Pr}_s[M(x;s) \neq D(x)] \;\leq\; \epsilon.$$

The probability that $M(x;s) = D(x)$ in time $c' \cdot h(x)^{1/\delta}$ is estimated by

$$\begin{aligned} &\mathbf{Pr}_s[M(x;s) = D(x) \text{ in time } c' \cdot h(x)^{1/\delta} \mid s \in \Gamma_M(x)] \\ &\quad = \quad 1 - \mathbf{Pr}_s[M(x;s) \neq D(x) \text{ in time } c' \cdot h(x)^{1/\delta} \mid s \in \Gamma_M(x)] \\ &\qquad\qquad -\mathbf{Pr}_s[\mathrm{Time}_M(x;s) > c' \cdot h(x)^{1/\delta} \mid s \in \Gamma_M(x)] \\ &\quad > \quad 1 - \epsilon - \frac{1}{c} \\ &\quad = \quad 1 - \epsilon - \frac{1-2\epsilon}{2} \quad = \quad \frac{1}{2}. \end{aligned}$$

By the definition of $\mathrm{Time}^*_M(x)$, we conclude that $\mathrm{Time}^*_M(x) \leq c' \cdot h(x)^{1/\delta}$, that is:

$$\mathrm{Time}^*_M(x) \leq c' \cdot E_s[\mathrm{Time}_M(x;s)^\delta \mid s \in \Gamma_M(x)]^{1/\delta}.$$

□

We finish this section by introducing the following terminology for probabilistic Turing machines.

Definition 2.4.5 (Time Complexity) A probabilistic Turing machine M is called *t-time bounded* (or a *t-time Turing machine*) if $\mathrm{Time}^*_M(x) \leq t(|x|)$ for all x. For a set $\mathcal{T}$ of functions, M is *$\mathcal{T}$-time bounded* (or a *$\mathcal{T}$-time machine*) if M is t-time bounded for some $t \in \mathcal{T}$.

2.5 Worst-Case Complexity Classes

For years, theoretical computer scientists have been interested in resource-bounded computations and have studied their complexity and structural properties. In this section, we shall review central concepts in the theory of worst-case complexity.

2.5.1 Computable Functions

One of the most natural concepts in worst-case complexity theory is "polynomial-time computability." A function f on Σ^* is *polynomial-time computable* (**P**-computable, for short) if there is a deterministic Turing machine with one input tape and one output tape (i.e., a transducer), which computes f in time polynomial in terms of length of the input. Denote by **FP** the collection of all polynomial-time computable functions.

To study algorithms on different objects, such as graphs, sequences, circuits, etc., we use an encoding of objects into strings; the encoding must be effective and secure. In this thesis, we use Regan's paring function [85] as the basis of encoding and extend the function to multi-functions. Formally, a *pairing function* is a bijection from $\Sigma^* \times \Sigma^*$ onto Σ^*.

First recall that the notation x^- denotes the predecessor of string x in the standard order on Σ^* unless x is the empty string.

Definition 2.5.1 (Paring Function) The function $\langle \cdot, \cdot \rangle$ from $\Sigma^* \times \Sigma^*$ to Σ^* is defined as follows: for all pairs $(x, y) \in \Sigma^* \times \Sigma^*$,

$$\langle x, y \rangle = \begin{cases} d(x)y & \text{if } |y| \leq 1, \\ d(x)i_2[(y^-)^-] & \text{otherwise,} \end{cases}$$

where $d(\lambda) = \lambda$, $d(0x) = 00d(x)$, $d(1x) = 11d(x)$, $i_2[0x] = 01x$, and $i_2[1x] = 10x$ for all x.

Below we list without proofs several important properties of this paring function:

(i) $\langle , \rangle$ is monotone, i.e., $x \leq x'$ and $y \leq y'$ imply $\langle x, y \rangle \leq \langle x', y' \rangle$.

(ii) $\langle , \rangle$ is computable in linear-time in the lengths of x and y.

(iii) $\langle 0^n, y \rangle > \langle 0^m, z \rangle$ if and only if $n + |y| = m + |z|$, $n > m$, or $y > z$.

(iv) $2|x| + |y| \leq |\langle x, y \rangle| \leq 2|x| + |y| + 1$ for all x and y.

(v) For any increasing function f on Σ^*, $\langle f(x), x \rangle^- \geq \langle f(x^-), x^- \rangle$ for all x.

This paring function is recursively generalized to a bijection from $(\Sigma^*)^k$ onto Σ^* as $\langle x_1, x_2, \cdots, x_k\rangle = \langle x_1, \langle x_2, \cdots, x_k\rangle\rangle$. It is clear that:

$$2\sum_{i=1}^{k-1}|x_i| + |x_k| \leq |\langle x_1, \ldots, x_k\rangle| \leq 2\sum_{i=1}^{k-1}|x_i| + |x_k| + 1.$$

2.5.2 Complexity Classes

A *complexity class* is a collection of subsets of Σ^*. For a complexity class $\mathcal{C}$, the *complement of* $\mathcal{C}$, denoted by co-$\mathcal{C}$, is the collection of sets S such that $\Sigma^* - S$ is in $\mathcal{C}$.

Definition 2.5.2 For a function t on $\mathbb{N}$, let DTIME(t), NTIME(t), and DSPACE(t) denote the class of all sets recognizable by deterministic t-time, nondeterministic t-time, and deterministic t-space Turing machines, respectively. Similarly, let BPTIME(t) denote the class of sets recognizable by t-time bounded probabilistic Turing machines with bounded error probability, and let RTIME(t) denote the collection of sets computable by one-sided error, probabilistic (i.e., random) t-time Turing machines. For a set $\mathcal{T}$ of functions, let DTIME($\mathcal{T}$) = $\bigcup_{t\in\mathcal{T}}$ DTIME(t). In a similar fashion, we define NTIME($\mathcal{T}$), BPTIME($\mathcal{T}$), and DSPACE($\mathcal{T}$).

Using these notations , we can introduce several important complexity classes. For deterministic classes, we use the following basic complexity classes:

1. **P** = DTIME($n^{O(1)}$) (polynomial-time).
2. **E** = DTIME($2^{O(n)}$) (linear-exponential-time).
3. **EXP** = DTIME($2^{n^{O(1)}}$) (exponential-time).

A set S in **P** is said to be *polynomial-time computable* (**P**-computable, for short). Similarly, we use the terminology *exponential-time computable* (**EXP**-computable, for short) and *linear-exponential-time computable* (**E**-computable, for short), respectively, for sets in **EXP** and in **E**.

For space-bounded complexity classes, we use:

1. **PSPACE** = DSPACE($n^{O(1)}$) (polynomial-space).
2. **ESPACE** = DSPACE($2^{O(n)}$) (linear-exponential-space).

It is worth noting that **PSPACE** is *closed under complement*, i.e., **PSPACE** = co-**PSPACE**.

For nondeterministic classes, we set:

1. **NP** = NTIME($n^{O(1)}$).
2. **NE** = NTIME($2^{O(n)}$).
3. **NEXP** = NTIME($2^{n^{O(1)}}$).

Interestingly, **NP**-sets have the following characterization by logical terms: a set A is **NP** if and only if there exists a set $B \in \mathbf{P}$ and a polynomial p such that $A = \{x \mid \exists y[|y| = p(|x|) \wedge \langle x, y\rangle \in B]\}$.

For probabilistic classes, let:

1. $\mathbf{RP} = \mathrm{RTIME}(n^{O(1)})$ (random polynomial-time).
2. $\mathbf{RE} = \mathrm{RTIME}(n^{O(1)})$ (random linear-exponential-time).
3. $\mathbf{BPP} = \mathrm{BPTIME}(n^{O(1)})$ (bounded-error probabilistic polynomial-time).
4. $\mathbf{BPE} = \mathrm{BPTIME}(2^{O(n)})$ (bounded-error probabilistic linear-exponential-time).

The class **ZPP** (zero-error probabilistic polynomial-time) is defined by $\mathbf{RP} \cap \text{co-}\mathbf{RP}$.

It is important to note that $\mathrm{DTIME}(O(n)) \neq \mathrm{NTIME}(O(n))$ [82]. This is the only separation result we have known until now.

Other important complexity classes are **UP** and **PP**. The class **UP** (unambiguous polynomial time) is defined by polynomial-time unambiguous Turing machines. The probabilistic class **PP** (probabilistic polynomial time) consists of sets which are defined by polynomial-time probabilistic Turing machines whose error probability is $< 1/2$. Actually, the value $1/2$ can be replaced by any real number ϵ satisfying $0 < \epsilon < 1$. By definition, $\mathbf{P} \subseteq \mathbf{UP} \subseteq \mathbf{NP}$ and $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$.

Using the notion of oracle Turing machines, we can introduce *relativized complexity classes*. For a set S and a complexity class $\mathcal{C}$, the notation $\mathcal{C}^S$ ($\mathcal{C}$ relative to S) denotes the class naturally obtained from the definition of $\mathcal{C}$ with the help of S as oracle, and $\mathcal{C}^{\mathcal{D}}$ ($\mathcal{C}$ relative to $\mathcal{D}$) denotes the union of all $\mathcal{C}^S$ for any $S \in \mathcal{D}$. All complexity classes defined above are naturally relativized using the notion of oracle Turing machines, such as $\mathbf{P}^A$, $\mathbf{NP}^A$, $\mathbf{BPP}^A$, $\mathbf{RP}^A$, $\mathbf{UP}^A$, etc.

As mentioned before, the empty oracle set does not change the computational power of the Turing machines; thus, we have $\mathbf{P}^{\emptyset} = \mathbf{P}$, $\mathbf{NP}^{\emptyset} = \mathbf{NP}$, etc. We remember an important result due to Zachos (see, e.g., [123]) that $\mathbf{BPP}^{\mathbf{BPP}} = \mathbf{BPP}$.

Let $\#\mathbf{P}$ (pronounced "sharp P" or "number P") be the collection of all functions on Σ^* which are computed by polynomial-time counting Turing machines. It is not difficult to see that f is in $\#\mathbf{P}$ if and only if there exist a set $A \in \mathbf{P}$ and a polynomial p such that $f(x) = \|\{y \in \Sigma^{p(|x|)} \mid \langle x, y\rangle \in A\}\|$ for all x.

The counting of solutions is relevant to probabilistic computation. The following result shows a relationship between these two notions.

Lemma 2.5.3 **[2]** $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}^{\mathbf{PP}}$ *and* $\mathbf{FP}^{\#\mathbf{P}} = \mathbf{FP}^{\mathbf{PP}}$.

Proof. First we show the inclusion $\mathbf{PP} \subseteq \mathbf{P}^{\#\mathbf{P}}$. This inclusion is easily seen as follows. For a set $A \in \mathbf{PP}$, there is a probabilistic Turing machine M. We modify M so that all computation paths of M on each input x are of the same length, say $p(x)$. The success probability of M on x is equal to the ratio of $\|\mathrm{Acc}(M, x)\|$ to $2^{p(x)}$. Let us define $f(x) = \|\mathrm{Acc}(M, x)\|$ for all x. Obviously $f \in \#\mathbf{P}$. Using this f, we can determine in polynomial time whether $2 \cdot f(x) > 2^{p(x)}$, which means $x \in A$. Therefore, A belongs to $\mathbf{P}^f \subseteq \mathbf{P}^{\#\mathbf{P}}$.

Next we shall show the other inclusion that $\#\mathbf{P} \subseteq \mathbf{FP}^{\mathbf{PP}}$; thus, we have $\mathbf{P}^{\#\mathbf{P}} \subseteq \mathbf{P}^{\mathbf{PP}}$.

Take a $\#\mathbf{P}$-function f. There exists a polynomial-time nondeterministic Turing machine M such that, for every x, $f(x)$ equals the number of accepting computation paths of M on input x. Take a polynomial p and assume that the number of nondeterministic computation paths of M on input x is exactly $2^{p(|x|)}$ (this is always true if we pad extra nondeterministic, rejecting paths).Clearly $0 \leq f(x) \leq 2^{p(|x|)}$ for all x.

We then define the set $A = \{\langle s_i, x\rangle \mid \|\mathrm{Acc}(M, x)\| \geq i\}$ so that:

$$f(x) = \max\{i \in \mathbb{N} \mid \langle s_i, x\rangle \in A\}.$$

A binary search technique helps us compute the value $f(x)$ deterministically by simply querying polynomially-many strings of the form $\langle s_i, x\rangle$, $0 \leq i \leq 2^{p(|x|)}$, to oracle A. Therefore, we have $f \in \mathbf{FP}^A$.

In the rest of the proof, we must show that A is in $\mathbf{PP}$. Let us define the following randomized Turing machine N:

begin randomized algorithm for N
  input $\langle s_i, x\rangle$
  if $i = 0$ **then accept** and **halt**
  generate a bit b ($b \in \{0, 1\}$) at random
  if $b = 0$ **then** simulate M on input x and **halt**
  generate a string y of length $p(|x|)$ at random
  (assume that y is the jth string in $\Sigma^{p(|x|)}$, where $0^{p(|x|)}$ is the 0th string)
  if $j < 2^{p(|x|)} - i$ **then accept else reject**
end.

This machine N obviously runs in polynomial time because M does so. It is also easy to see that, for each $\langle s_i, x\rangle$,

$$\langle s_i, x\rangle \in A \Longleftrightarrow \mathbf{Pr}_N[N(\langle s_i, x\rangle) = 1] > \frac{1}{2}.$$

Hence, $A \in \mathbf{PP}$. □

A set S is in $\oplus\mathbf{P}$ (pronounced "parity P") if there exist a polynomial p and a set $A \in \mathbf{P}$ such that, for each x, $x \in S$ if and only if $\|\{y \in \Sigma^{p(|x|)} \mid \langle x, y\rangle \in A\}\|$ is odd [81]. We remark that $\mathbf{UP} \subseteq \oplus\mathbf{P}$. A set S is *near-testable* if there exists an $f \in \mathbf{FP}$ such that, for all nonempty strings x, $f(x) \equiv \chi_S(x) + \chi_S(x^-) \pmod 2$. Let $\mathbf{NT}$ denote the collection of all near-testable sets [31]. It is known that $\mathbf{P} \subseteq \mathbf{NT} \subseteq \oplus\mathbf{P}$ and $\mathbf{NT}$ is also included in $\mathbf{E} \cap \mathbf{PSPACE}$ [31]. It is also known that $\oplus\mathbf{P}^{\oplus\mathbf{P}} = \oplus\mathbf{P}$ [18] and $\mathbf{NP} \subseteq \mathbf{RP}^{\oplus\mathbf{P}}$ [105].

The class $\mathbf{P}/\mathrm{poly}$ consists of all sets A such that there exist a p-bounded function f from $\mathbb{N}$ to Σ^* and a set $B \in \mathbf{P}$ satisfying $A = \{x \mid \langle x, f(|x|)\rangle \in B\}$. This class is also known as the collection of all sets computable by (non-uniform) families of polynomial-size circuits.

A set S is in $\mathbf{APT}$ (*almost polynomial time*) [71] if there exists a polynomial p and a deterministic Turing machine M which accepts S such that the set $\{x \mid \mathrm{Time}_M(x) \geq p(|x|)\}$ is sparse.

2.5.3 Worst-Case Hierarchies

We shall define several important hierarchies in worst-case complexity theory.

Definition 2.5.4 (Polynomial Time Hierarchy) [72, 101, 118] The *polynomial-time hierarchy* consists of the following complexity classes: $\mathbf{\Delta}_0^{\mathrm{p}} = \mathbf{\Sigma}_0^{\mathrm{p}} = \mathbf{\Pi}_0^{\mathrm{p}} = \mathbf{P}$; $\mathbf{\Delta}_k^{\mathrm{p}} = \mathbf{P}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}}$; $\mathbf{\Sigma}_k^{\mathrm{p}} = \mathbf{NP}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}}$; and $\mathbf{\Pi}_k^{\mathrm{p}} = \text{co-}\mathbf{\Sigma}_k^{\mathrm{p}}$ for $k > 0$. We also use the cumulative polynomial-time hierarchy $\mathbf{PH} = \bigcup_{k \geq 0} \mathbf{\Delta}_k^{\mathrm{p}}$.

Note that $\mathbf{\Sigma}_k^{\mathrm{p}} \cup \mathbf{\Pi}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_{k+1}^{\mathrm{p}} \subseteq \mathbf{\Sigma}_{k+1}^{\mathrm{p}} \cap \mathbf{\Pi}_{k+1}^{\mathrm{p}}$ for all $k > 0$ (see [101]). It is known that if $\mathbf{NP} \subseteq \mathbf{BPP}$, then $\mathbf{NP} = \mathbf{RP}$ [52] and also $\mathbf{PH} = \mathbf{BPP}$ [123]. A recent achievement is Toda's theorem [102] that $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{PP}}$.

For a function f on $\mathbb{N}$ and a set A, let $\mathbf{P}^{A[O(f(n))]}$ be the collection of sets B which are computed by a polynomial-time deterministic oracle Turing machine M with oracle A such that the number of queries by M on input x is bounded above by $c \cdot f(n) + d$, where c and d are constants depending only on M. For a class $\mathcal{C}$ of sets, set $\mathbf{P}^{\mathcal{C}[O(f(n))]}$ to be the union of all $\mathbf{P}^{A[O(f(n))]}$ for every $A \in \mathcal{C}$. In particular, we write $\mathbf{\Theta}_k^{\mathrm{p}}$ for $\mathbf{P}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}[O(\log n)]}$.

Moreover, we define two hierarchies over $\mathbf{E}$ and $\mathbf{EXP}$ as follows.

Definition 2.5.5 (Exponential-Time Hierarchies) cf. [4]

1. The *linear exponential-time hierarchy* is defined as follows: $\mathbf{\Delta}_0^{\mathrm{e}} = \mathbf{\Sigma}_0^{\mathrm{e}} = \mathbf{\Pi}_0^{\mathrm{e}} = \mathbf{E}$, and for each $k > 0$, $\mathbf{\Delta}_k^{\mathrm{e}} = \mathbf{E}^{\mathbf{\Sigma}_k^{\mathrm{p}}}$, $\mathbf{\Sigma}_k^{\mathrm{e}} = \mathbf{NE}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}}$, and $\mathbf{\Pi}_k^{\mathrm{e}} = \text{co-}\mathbf{\Sigma}_k^{\mathrm{e}}$. Let $\mathbf{EH} = \bigcup_{k \geq 0} \mathbf{\Delta}_k^{\mathrm{e}}$.

2. The *exponential-time hierarchy* is defined as follows: $\mathbf{\Delta}_0^{\exp} = \mathbf{\Sigma}_0^{\exp} = \mathbf{\Pi}_0^{\exp} = \mathbf{EXP}$, and for each $k > 0$, $\mathbf{\Delta}_k^{\exp} = \mathbf{EXP}^{\mathbf{\Sigma}_k^{\mathrm{p}}}$, $\mathbf{\Sigma}_k^{\exp} = \mathbf{NEXP}^{\mathbf{\Sigma}_k^{\mathrm{p}}}$, and $\mathbf{\Pi}_k^{\exp} = \text{co-}\mathbf{\Sigma}_k^{\exp}$. Let $\mathbf{EXPH} = \bigcup_{k \geq 0} \mathbf{\Delta}_k^{\exp}$.

We next consider alternating Turing machines.

Definition 2.5.6 Let $\mathrm{ATIME}(t(n))$ be the class of all sets which are computed by alternating Turing machines in time $t(n)$. We also define two alternation-bounded classes. Let $\mathrm{ATIME}^{\Delta}(k(n), t(n))$ be the class of all sets computed by semi-deterministic alternating Turing machines with at most $k(n)$-alternations in time $t(n)$. Similarly, let $\mathrm{ATIME}^{\Sigma}(k(n), t(n))$ be the class of all sets computed by alternating Turing machines, starting with existential states, with at most $k(n)$ alternations and in time $t(n)$. For sets $\mathcal{K}$ and $\mathcal{T}$ of functions, we define $\mathrm{ATIME}^{\Sigma}(\mathcal{K}, \mathcal{T}) = \bigcup_{k \in \mathcal{K}} \bigcup_{t \in \mathcal{T}} \mathrm{ATIME}^{\Sigma}(k(n), t(n))$, and $\mathrm{ATIME}^{\Delta}(\mathcal{F}, \mathcal{T})$ and $\mathrm{ATIME}(\mathcal{T})$ can be defined similarly.

All three hierarchies introduced above are characterized by alternating Turing machines with constant-alternation.

Lemma 2.5.7 *Let $k > 0$. Then,*

1. $\mathbf{\Delta}_k^{\mathrm{p}} = \mathrm{ATIME}^{\Delta}(k, n^{O(1)})$, $\mathbf{\Sigma}_k^{\mathrm{p}} = \mathrm{ATIME}^{\Sigma}(k, n^{O(1)})$, *and* $\mathbf{PSPACE} = \mathrm{ATIME}(n^{O(1)})$.

2. $\mathbf{\Delta}_k^{\mathrm{e}} = \mathrm{ATIME}^{\Delta}(k, 2^{O(n)})$ *and* $\mathbf{\Sigma}_k^{\mathrm{e}} = \mathrm{ATIME}^{\Sigma}(k, 2^{O(n)})$.

3. $\mathbf{\Delta}_k^{\exp} = \mathrm{ATIME}^{\Delta}(k, 2^{n^{O(1)}})$ *and* $\mathbf{\Sigma}_k^{\exp} = \mathrm{ATIME}^{\Sigma}(k, 2^{n^{O(1)}})$.

Proof. Here we show only the claim for $\mathbf{\Delta}_k^{\mathrm{p}}$. For a set $A \in \mathbf{\Delta}_k^{\mathrm{p}}$, take a deterministic oracle Turing machine M which is p-time bounded and an oracle set $B \in \mathbf{\Sigma}_{k-1}^{\mathrm{p}}$ (if $k = 1$, then let $B = \emptyset$) such that $A = L(M, B)$, where p is an increasing polynomial. Since $B \in \mathbf{\Sigma}_{k-1}^{\mathrm{p}}$, there are k-alternation bounded polynomial-time alternating Turing machines N_0 and N_1 which recognize $\overline{B}$ and B, respectively. Now we construct a semi-deterministic alternating Turing machine which recognizes A as follows: on input x, simulate M on input x except for oracle queries; whenever M queries y_i, guess its oracle answer $ans(y_i)$ and store values y_i and $ans(y_i)$; then universally choose $\lfloor \log p(n) \rfloor$ bits c, and if c is the ith element of $\Sigma^{\lfloor \log(p(n)) \rfloor}$, then erase all symbols except y_i and $ans(y_i)$, and finally simulate N_0 on y_i if $ans(y_i) = 0$, or else, simulate N_1 on y_i.

Conversely, take a polynomial-time semi-deterministic alternating Turing machine M. We build a deterministic oracle machine M' and an oracle set B such that $L(M) = L(M, B)$. Step by step, we decide a computation path of M on input x which leads to an accepting configuration of M on x if one exists. Assume that $b_1, \cdots, b_m$ are determined, and M' is at the $(m+1)$th node c which has two children c_0, c_1. To decide which child node to choose, we follow the procedure: follow the leftmost path until M enters a universal state, and then choose the path labeled with the $(m+1)$th string; letting p be a label of this path, query a string $\langle x, p \rangle$; if the oracle answers "yes," then let $b_{m+1} = 1$, and otherwise, let $b_{m+1} = 0$; then choose $c_{b_{m+1}}$ and go into the next node.

Oracle set B is defined by the following algorithm: on input $\langle x, p \rangle$, simulate M on x and deterministically follow a computation path labeled with p and then simulate the rest of the computation tree T. Since T has $(k-1)$-alternation, B belongs to $\mathrm{ATIME}^{\Sigma}(k-1, n^{O(1)}) = \mathbf{\Sigma}_{k-1}^{\mathrm{p}}$. Clearly, we have $L(M) = L(M', B)$. □

The relationship between the polynomial-time hierarchy and the linear exponential-time alternation hierarchy is summarized as the following lemma.

Lemma 2.5.8 *For each $k > 0$,* $\mathrm{TALLY} \cap \mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}$ *if and only if* $\mathbf{\Delta}_k^{\mathrm{e}} = \mathbf{\Sigma}_k^{\mathrm{e}}$.

Proof. Use Book's tally-encoding technique [14]. For a set A, define a tally part of A as $Tally(A) = \{0^n \mid \text{the binary representation of } n \text{ is of the form } 1w \text{ and } w \in A\}$. It is not difficult to see that $A \in \mathrm{ATIME}^{\Sigma}(k, 2^{O(n)})$ if and only if $Tally(A) \in \mathrm{ATIME}^{\Sigma}(k, n^{O(1)})$. Hence, $A \in \mathbf{\Sigma}_k^{\mathrm{e}}$ if and only if $Tally(A) \in \mathbf{\Sigma}_k^{\mathrm{p}}$. A similar equivalence relation also holds between $\mathbf{\Delta}_k^{\mathrm{e}}$ and $\mathbf{\Delta}_k^{\mathrm{p}}$. The lemma, therefore, follows from these characterizations. □

Schöning [89] has constructed two hierarchies within **NP**.

Definition 2.5.9 (Low and High Hierarchies within NP) [89] Let $n \geq 1$.

1. The *low hierarchy* within **NP** is defined as follows: $\mathbf{L}\boldsymbol{\Delta}_n^{\mathrm{p}} = \{A \in \mathbf{NP} \mid \boldsymbol{\Delta}_n^{\mathrm{p}}(A) \subseteq \boldsymbol{\Delta}_n^{\mathrm{p}}\}$ and $\mathbf{L}\boldsymbol{\Sigma}_n^{\mathrm{p}} = \{A \in \mathbf{NP} \mid \boldsymbol{\Sigma}_n^{\mathrm{p}}(A) \subseteq \boldsymbol{\Sigma}_n^{\mathrm{p}}\}$.

2. The *high hierarchy* within **NP** is defined as follows: $\mathbf{H}\boldsymbol{\Delta}_n^{\mathrm{p}} = \{A \in \mathbf{NP} \mid \boldsymbol{\Delta}_{n+1}^{\mathrm{p}} \subseteq \boldsymbol{\Delta}_n^{\mathrm{p}}(A)\}$ and $\mathbf{H}\boldsymbol{\Sigma}_n^{\mathrm{p}} = \{A \in \mathbf{NP} \mid \boldsymbol{\Sigma}_{n+1}^{\mathrm{p}} \subseteq \boldsymbol{\Sigma}_n^{\mathrm{p}}(A)\}$.

2.5.4 Polynomial-Time Reducibilities

Polynomial-time reducibilities play a very important role in computational complexity theory. We briefly sketch such reducibilities.

A set D is called *polynomial-time many-one reducible* (p-m-reducible, for short) to a set E, denoted by $D \leq_m^{\mathrm{p}} E$, if there exists a function f in **FP** such that, for all x, $x \in D$ if and only if $f(x) \in E$. This function f is called a *(polynomial-time many-one) reduction* and is said to *reduce D to E*. Furthermore, if f is one-one, then we say that D is *polynomial-time 1-1 reducible* (p-1-reducible, for short) to E. A set D is *polynomial-time Turing reducible* (p-T-reducible, for short) to E, denoted by $D \leq_T^{\mathrm{p}} E$, if there exists a deterministic polynomial-time Turing machine M such that $D = L(M, E)$. A set D is *polynomial-time truth-table reducible* (p-tt-reducible, for short) to E, denoted by $D \leq_{tt}^{\mathrm{p}} E$, if there is a polynomial-time oracle Turing machine M which with oracle E makes nonadaptive queries such that $D = L(M, E)$.

A set D is *polynomial-time many-one (Turing, truth-table,* resp.*) complete*, p-m-complete, for short, (p-T-complete, p-tt-complete, resp., for short) for a class $\mathcal{C}$ if $D \in \mathcal{C}$ and every set in $\mathcal{C}$ is p-m-reducible (p-T-reducible, p-tt-reducible, resp.) to D. For a complexity class $\mathcal{C}$, we simply say that D is *$\mathcal{C}$-complete* if D is p-m-complete for $\mathcal{C}$.

One of the most useful **NP**-complete problems is the *bounded halting problem*, BHP, defined as follows: assuming that $\{M_i\}_{i \in \mathbb{N}}$ is an effective enumeration of all nondeterministic polynomial-time Turing machines, let

$$\mathrm{BHP} = \{\langle s_i, x, 1^n \rangle \mid M_i \text{ accepts } x \text{ within time } n\ \}.$$

We quickly sketch the proof that BHP is **NP**-complete. To see that BHP $\in$ **NP**, it is enough to check the following algorithm:

begin nondeterministic algorithm for BHP
  input y
  if y is not of the form $\langle s_i, x, 1^n \rangle$ **then** reject
  (Now assume that $y = \langle s_i, x, 1^n \rangle$)
  simulate nondeterministically M_i on input x for n steps
  if M_i does not halt **then** reject
end.

The running time of this algorithm is bounded by a polynomial in n since each simulation of machine M_i does not exceed n steps. Hence, BHP $\in$ **NP**. Next we show that BHP is **NP**-hard. For any **NP** set A, take a nondeterministic Turing machine M which recognizes A in polynomial time. Also take a strictly increasing

polynomial p such that $\mathrm{Time}_M(x) \leq p(|x|)$ for all strings x. Let i be an index such that $L(M) = L(M_i)$. Now let us define f as $f(x) = \langle s_i, x, 1^{p(|x|)} \rangle$. The function f reduces A to BHP and is clearly one-one, p-honest, increasing, length-preserving, and **P**-computable.

Notice from the above proof that every **NP** set is p-1-reducible to BHP by p-honest, monotone, length-preserving reductions.

Another typical example of **NP**-complete sets is the *satisfiability problem* SAT that is defined as

$$\mathrm{SAT} = \{\langle F \rangle \mid F \text{ is a satisfiable formula } \},$$

where $\langle F \rangle$ denotes an appropriate binary encoding of a formula F. There is a p-honest, one-one reduction from BHP to SAT; therefore, SAT is also **NP**-complete.

The *Turing closure* (*many-one closure, truth-table closure*, resp.) of a class $\mathcal{C}$ is the collection of sets which are polynomial-time Turing (many-one, truth-table, resp.) reducible to some sets in $\mathcal{C}$.

Lemma 2.5.10 *Every set in $\mathbf{\Delta}_k^{\mathrm{exp}}$, $k > 0$, is p-m-reducible to some set in $\mathbf{\Delta}_k^{\mathrm{e}}$. That is, the many-one closure of $\mathbf{\Delta}_k^{\mathrm{e}}$ is exactly $\mathbf{\Delta}_k^{\mathrm{exp}}$.*

Proof. By a padding argument. Assume that A is a set in $\mathbf{\Delta}_k^{\mathrm{exp}}$. There exist a polynomial p and a semi-deterministic alternating Turing machine M which, on input x, recognizes A in time $2^{p(|x|)}$ with k-alternations. Let N be another machine that, on input x of the form $x01^{p(|x|)}$, simulates M on input x. By definition, it follows that $N(x01^{p(|x|)}) = M(x)$ for all x. It is important to notice that N is $2^{O(n)}$-time bounded. For the desired reduction, define $f(x) = x01^{p(|x|)}$ for all x. □

Two sets A and B are called *polynomially isomorphic* (p-isomorphic, for short) if there exists a **P**-computable, p-invertible bijection f which reduces A to B. This reduction f is called a *polynomial-time isomorphism* (p-isomorphism, for short). Berman and Hartmanis [10] raised the question of whether **NP**-complete sets are all p-isomorphic. This is known as the "isomorphism conjecture."

A set S is **P**-*printable* if there is a **P**-computable function f such that $f(0^n)$ outputs the list of all strings in $S \cap \Sigma^n$. It is known that a set A is **P**-printable if and only if A is p-isomorphic to some tally set [1]. It is easy to see that if $\mathbf{P} = \mathbf{NP}$, then all sparse sets in **P** are **P**-printable [1].

A set S is *(Turing) self-reducible* if there exists a deterministic oracle Turing machine M such that:

(i) $S = L(M, S)$; and

(ii) On every input x, M queries only strings whose length is smaller than $|x|$.

For example, SAT and BHP are both self-reducible.

2.5.5 Complexity Cores

We shall review a notion of *complexity cores* and their existence shown by Book and Du [16].

Definition 2.5.11 (Complexity Cores) Let A be a set and let $\mathcal{C}$ be a class of sets. An infinite set H is a *complexity core (*or *hard core) for A with respect to $\mathcal{C}$* if, for every set C in $\mathcal{C}$, if $C \subseteq A$, then $C \cap H$ is finite. A complexity core H is called *proper* if $H \subseteq A$.

Lemma 2.5.12 **[16]** *Let $\mathcal{C}$ be a recursively enumerable class of recursive sets. If $\mathcal{C}$ is closed under finite union and finite variation, then any infinite recursive set A not in $\mathcal{C}$ has an infinite recursive, proper complexity core for A with respect to $\mathcal{C}$.*

Proof. Assume that A is an infinite recursive set not in $\mathcal{C}$. Since $\mathcal{C}$ is recursively enumerable, all subsets of A that are in $\mathcal{C}$ can be effectively enumerated as $\{C_0, C_1, \ldots\}$. For each $k \in \mathbb{N}$, let $D_k = \bigcup_{i=0}^{k} C_i$. Note that $D_j \subseteq D_{j+1}$ for every j. Now let $D = \bigcup_{j \geq 0} D_j$. If $A - D$ is infinite, then $A - D$ is an infinite proper complexity core for A since any subset C of A that is in $\mathcal{C}$ is a subset of D, and thus $(A_D) \cap C = \emptyset$. Now let us assume that $A - D$ is finite. There are infinitely many k satisfying that $D_k \neq D_{k+1}$ since, otherwise, $\mathcal{C}$ is closed under finite union, and thus A is in $\mathcal{C}$, a contradiction. For each k with $D_k \neq D_{k+1}$, take the element a_k that is the minimal in $D_{k+1} - D_k$. Let $H = \{a_k \mid D_k \neq D_{k+1}\}$. This H is clearly infinite and also a proper complexity core for A. □

2.6 One-Way Functions

This section will define *one-way functions*. A one-way function is a function which is computed easily but whose inverse is hard to compute. We shall introduce the new notions of *nearly-***RP** and *nearly-***BPP** sets and show that if one-way functions exist, then all **NP** sets are nearly-**BPP** using *hash function* technique.

2.6.1 Hash Functions

We shall introduce *hash functions* as a useful tool in the discussion of randomized algorithms.

For $n, m \in \mathbb{N}$ $(n < m)$, let $H_{n,m}$ denote the family of pairwise independent *universal hash functions* from Σ^n to Σ^m which is defined as follows: a hash function h in $H_{n,m}$ is of the form $h = (M, b)$, where M is an m by n bit matrix and b is a bit vector, and takes its value as $h(x) = Mx \oplus b$. Hence, the set $H_{n,m}$ can be identified with the set of all m by $n+1$ matrices over $\{0,1\}$, and each hash function h is encoded into a string of length $m(n+1)$. Note that $\|H_{n,m}\| = 2^{m(n+1)}$.

Lemma 2.6.1 **[20]** *If $x \neq y$, $n < m$, and $i \leq m$, then $\mathbf{Pr}_h[h(x)_{\leftarrow i} = h(y)_{\leftarrow i} \mid h \in H_{n,m}] = 2^{-i}$. Moreover $\mathbf{Pr}_h[h(x)_{\leftarrow i} = w_{\leftarrow i} \mid h \in H_{n,m}] = 2^{-i}$ for fixed x and w with $|w| \geq i$.*

Proof. An easy exercise. □

Fix n and c and assume $i \leq n$ and $\|X\| > 0$. We say that a function h from Σ^n to Σ^{n+c} *i-distinguishes*

x on X if $h(x)_{\leftarrow i+c} \neq h(w)_{\leftarrow i+c}$ for all $w \in X - \{x\}$; otherwise, *h i-indistinguishes x on X*.

Proposition 2.6.2 *Let $n \in \mathbb{N}$, $x \in \Sigma^*$, and $i \in \mathbb{N}$. Assume that* $\mathrm{ilog}(\|X\|) \leq i \leq n$.

1. $\mathbf{Pr}_h[h \text{ } i\text{-distinguishes } x \text{ on } X \mid h \in H_{n,n+c}] \geq 1 - 2^{-c}$.
2. $(1-2^{-c})2^{-i-c} \leq \mathbf{Pr}_{hy}[h(x)_{\leftarrow i+c} = y_{\leftarrow i+c} \wedge h \text{ } i\text{-distinguishes } x \text{ on } X \mid h \in H_{n,n+c} \wedge y \in \Sigma^{n+c}] \leq 2^{-i-c}$.

Proof. 1) Fix x and i, and let $\rho_{i,x} = \mathbf{Pr}_h[h \text{ } i\text{-distinguishes } x \text{ on } X \mid h \in H_{n,n+c}]$. Then,

$$\begin{aligned}
\rho_{i,x} &= 1 - \mathbf{Pr}_h[\exists w \in X - \{x\}(h(w)_{\leftarrow i+c} = h(x)_{\leftarrow i+c}) \mid h \in H_{n,n+c}] \\
&= 1 - \sum_{w \in X-\{x\}} \mathbf{Pr}_h[h(w)_{\leftarrow i+c} = h(x)_{\leftarrow i+c} \mid h \in H_{n,n+c}] \\
&= 1 - \frac{\|X\| - 1}{2^{i+c}} \quad \geq \quad 1 - \frac{\|X\| - 1}{2^{\log(\|X\|)+c}} \\
&= 1 - \frac{\|X\| - 1}{\|X\| \cdot 2^c} \quad \geq \quad 1 - \frac{1}{2^c}.
\end{aligned}$$

2) Let $\rho'_{i,x} = \mathbf{Pr}_{hy}[h(x)_{\leftarrow i+c} = y_{\leftarrow i+c} \wedge h \text{ } i\text{-distinguishes } x \text{ on } X \mid h \in H_{n,n+c} \wedge y \in \Sigma^{n+c}]$. We show the first inequality of the claim:

$$\begin{aligned}
\rho'_{i,x} &\leq \mathbf{Pr}_{hy}[h(x)_{\leftarrow i+c} = y_{\leftarrow i+c} \mid h \in H_{n,n+c}, y \in \Sigma^{n+c}] \\
&\leq \sum_{y:|y|=n+c} 2^{-(n+c)} \cdot \mathbf{Pr}_h[h(x)_{\leftarrow i+c} = y_{\leftarrow i+c} \mid h \in H_{n,n+c}] \\
&= \sum_{y:|y|=n+c} 2^{-(n+c)} \cdot 2^{-(i+c)} \quad = \quad 2^{-(i+c)}.
\end{aligned}$$

The second inequality is shown as follows. In the calculation, we omit the term "$h \in H_{n,n+c}$."

$$\begin{aligned}
\rho'_{i,x} &= \sum_{y:|y|=n+c} 2^{-(n+c)} \cdot \mathbf{Pr}_h[h(x)_{\leftarrow i+c} = y_{\leftarrow i+c}] \\
&\qquad \times \mathbf{Pr}_h[h \text{ } i\text{-distinguishes } x \text{ on } X \mid h(x)_{\leftarrow i+c} = y_{\leftarrow i+c}] \\
&= \sum_{y:|y|=n+c} 2^{-(n+c)} \cdot 2^{-(i+c)} \cdot \left(1 - \sum_{w \in X-\{x\}} \mathbf{Pr}_h[h(w)_{\leftarrow i+c} = y_{\leftarrow i+c} \mid h(x)_{\leftarrow i+c} = y_{\leftarrow i+c}]\right) \\
&\geq \sum_{y:|y|=n+c} 2^{-(n+c)} \cdot 2^{-(i+c)} \cdot \left(1 - \frac{\|X\| - 1}{2^{-(i+c)}}\right) \\
&\geq 2^{-(i+c)} \cdot \left(1 - \frac{\|X\|}{2^{\log(\|X\|)+c}}\right) \quad \geq \quad 2^{-(i+c)}(1 - 2^{-c}).
\end{aligned}$$

□

2.6.2 One-Way Functions

In computational complexity theory, there are several definitions of *one-way functions*. A function f is *polynomially invertible* (p-invertible, for short) if there is a function g in **FP** such that $g \circ f(x) = x$ for

all x, whereas f is *weakly p-invertible* if there is a **P**-computable function g such that $f \circ g(x) = x$ for all $x \in \mathrm{ran}(f)$. Note that if f is weakly p-invertible, then we can determine whether $x \in \mathrm{ran}(f)$ by checking if $f \circ g(x) = x$.

A *(weakly) one-way function* is a one-one, p-honest, P-computable function on Σ^* whose inverse is not computable in polynomial-time (cf. [80, 45, 4]). It is shown in [33, 53] that one-way functions exist if and only if $\mathbf{P} \neq \mathbf{UP}$.

In cryptography, slightly different one-way functions are used. We need only uniform one-wayness in this thesis. A *(uniform) strong one-way function* is a function f such that f is **P**-computable and, for all randomized Turing machines M working in polynomial-time, the function

$$\lambda n.\mathbf{Pr}_{x,s}[f(M(f(x) : s)) = f(x) \mid x \in \Sigma^n \wedge s \in \Omega_M(x)]$$

is negligible. Håstad, Impagliazzo, Levin, and Luby [40] showed a close relationship between the existence of strong one-way functions and that of pseudo-random (number) generators.

Let us introduce **RP**-like and **BPP**-like sets, called *nearly*-**RP** and *nearly*-**BPP** sets, respectively, which look like one-sided and two-sided, bounded-error probabilistic sets on most instances.

Definition 2.6.3 (Nearly-BPP Sets and Nearly-RP Sets) [119]

1. A set A is *nearly*-**BPP** if, for every polynomial p, there exist a set S and a polynomial-time randomized Turing machine M such that, for each x,

   (i) $x \in \Sigma^* - S$ implies $\mathbf{Pr}_M[M(x) \neq A(x)] \leq \frac{1}{3}$; and

   (ii) $\mathbf{Pr}_x[x \in S \mid x \in \Sigma^n] < \frac{1}{p(n)}$ for almost all n.

2. A set A is *nearly*-**RP** if, for every polynomial p, there exist a set S and a polynomial-time randomized Turing machine M such that, for each x,

   (i) $x \in A - S$ implies $\mathbf{Pr}_M[M(x) \neq A(x)] \leq \frac{1}{2}$;

   (ii) $x \in \overline{A} - S$ implies $\mathbf{Pr}_M[M(x) \neq A(x)] = 0$; and

   (iii) $\mathbf{Pr}_x[x \in S \mid x \in \Sigma^n] < \frac{1}{p(n)}$ for almost all n.

We can amplify the success probability of M on string inputs from $A - S$ by repeating its computations at random. A new Turing machine N is defined as follows: on input x ($n = |x|$), repeatedly run M on input x independently $p(n)$ many times, and accept x if and only if $M(x) = 1$ for some trial. Consider x in $A - S$. Then, the error probability that $N(x) = 0$ is at most $(\frac{1}{2})^{p(n)}$. Hence, $\mathbf{Pr}_N[N(x) = A(x)] \geq 1 - 2^{-p(n)}$. On the other hand, if $x \in \overline{A} - S$, then $\mathbf{Pr}_N[N(x) \neq A(x)] = 0$. Without loss of generality, we can further assume that the length of all nondeterministic computation paths of N on x is exactly $p(|x|)$ for some polynomial p.

Clearly, from the definition, **RP** (**BPP**, resp.) is properly contained in the class of nearly-**RP** (nearly-**BPP**, resp.) sets.

Here we show that the assumption that every **NP** set is nearly-**RP** implies that no strong one-way functions exist.

Proposition 2.6.4 **[119]** *If every* **NP** *set is nearly-***BPP***, then there is no strong one-way function.*

Proof. Assume that every **NP** set is nearly-**BPP** and a strong one-way function exists. By [9], there exists a length-preserving strong one-way function which is one-one on at least $\frac{2^n}{p(n)}$ elements in Σ^n for each n, where p is an adequate increasing polynomial. Let f be such a function. Let $D = \{x \mid \|f^{-1}(x)\| = 1\}$. We then have $\|D \cap \Sigma^n\| \geq \frac{2^n}{p(n)}$ for almost all n.

Denote by s_i^n the ith string of $\Sigma^{\mathrm{ilog}(n)}$ (in particular, $s_0^n = 0^{\mathrm{ilog}(n)}$). Consider the following sets $\{T_x\}_x$: for each x,

$$T_x = \{xs_i^n \mid \exists z \in \Sigma^{|x|}[f(z) = x \wedge \text{ the } (i+1)\text{-th bit of } z \text{ is } 1,\ 0 \leq i < n\ ]\}.$$

Note that $T_x \cap T_y = \emptyset$ unless $x = y$. Let $A = \bigcup_x T_x$. Clearly, A is in **NP**. By our assumption, for the polynomial $4n^2p(n)$, there is a set S and a polynomial-time randomized Turing machine M satisfying conditions (i)-(ii) of Definition 2.6.3(1) with $\|S^n\| < \frac{2^n}{4n^2 \cdot p(n)}$ for almost all n. In particular:

$$\|S^{n+\mathrm{ilog}(n)}\| \leq \frac{2^{n+\mathrm{ilog}(n)}}{4(n+\mathrm{ilog}(n))^2 p(n+\mathrm{ilog}(n))} \leq \frac{2n \cdot 2^n}{4n^2 \cdot p(n)} = \frac{2^n}{2n \cdot p(n)}.$$

Hence, we have:

$$\begin{aligned}
\mathbf{Pr}_x[T_x \not\subseteq A - S \mid x \in \Sigma^n] &= \mathbf{Pr}_x[\exists i[xs_i^n \in A \cap S] \mid x \in \Sigma^n] \\
&\leq \sum_{i=0}^{n-1} \mathbf{Pr}_x[xs_i^n \in A \cap S \mid x \in \Sigma^n] \\
&\leq \sum_{i=0}^{n-1} \mathbf{Pr}_x[xs_i^n \in S \mid x \in \Sigma^n] \ \leq \ \frac{n \cdot \|S^{n+\mathrm{ilog}(n)}\|}{2^n} \\
&< \frac{n}{2^n} \cdot \frac{2^n}{2n \cdot p(n)} \ = \ \frac{1}{2p(n)}.
\end{aligned}$$

Hence, $\mathbf{Pr}_x[T_x \subseteq A - S \mid x \in \Sigma^n] \geq 1 - \frac{1}{2p(n)}$. Moreover:

$$\mathbf{Pr}_x[x \in D \wedge T_x \subseteq A - S \mid x \in \Sigma^n] \geq \mathbf{Pr}_x[x \in D \mid x \in \Sigma^n] + \mathbf{Pr}_x[T_x \subseteq A - S \mid x \in \Sigma^n] - 1 \geq \frac{1}{2p(n)}.$$

We assume that, for $x \notin S$, $\mathbf{Pr}_M[M(x) = A(x)] \geq 1 - 2^{-|x|}$. We then define the randomized Turing machine N as follows:

begin randomized algorithm for N
  input x (say, $n = |x|$)
  let $z := \lambda$
  for $i = 0$ **to** $n - 1$
    if $M(xs_i^n) = 1$ **then** let $z := z1$ **else** let $z := z0$
  end-for

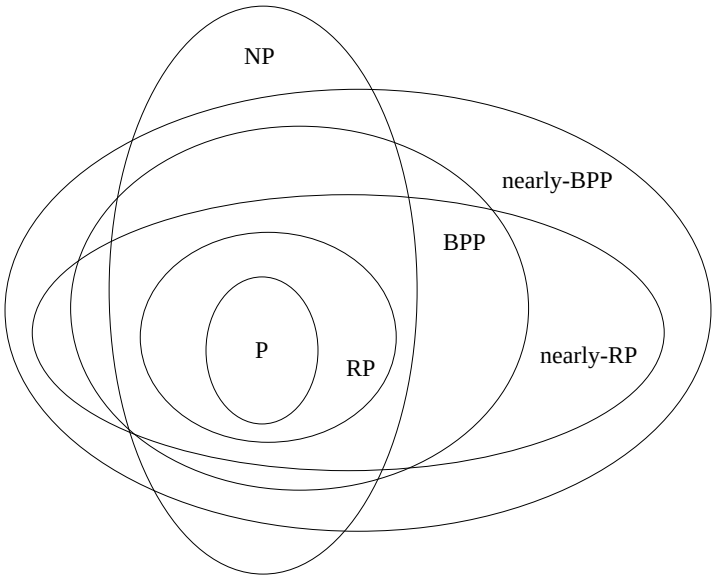

Figure 2.2: Predicted inclusion relationships

output z

end.

Assume that $x \in D$ and $T_x \subseteq A - S$. The success probability $\mathbf{Pr}_N[f(N(x)) = x]$ is bounded by

$$\mathbf{Pr}_N[f(N(x)) = x] \geq \mathbf{Pr}_M[M(xs_i^n) = A(xs_i^n)] \geq (1 - 2^{-n-\mathrm{ilog}(n)})^n \geq 1 - 2^{-\mathrm{ilog}(n)-1} > \frac{1}{2}.$$

Thus, we obtain $\mathbf{Pr}_N[f(N(x)) = x] > 1/2$. Using this inequality, the probability that N computes the inverse of f is bounded by

$$\begin{aligned}
&\mathbf{Pr}_{x,s}[f(N(f(x);s)) = f(x) \mid x \in \Sigma^n \wedge s \in \Omega_M(x)] \\
&\quad \geq \quad \mathbf{Pr}_x[x \in D \wedge T_x \subseteq A - S \mid x \in \Sigma^n] \\
&\qquad\qquad \times \mathbf{Pr}_{x,s}[f(N(x;s)) = x \mid x \in D \cap \Sigma^n \wedge T_x \subseteq A - S \wedge s \in \Omega_M(x)] \\
&\quad \geq \quad \frac{1}{2p(n)} \cdot \mathbf{Pr}_N[f(N(x)) = x] \\
&\quad \geq \quad \frac{1}{2p(n)} \cdot \frac{1}{2} \quad = \quad \frac{1}{4p(n)}.
\end{aligned}$$

This contradicts the one-wayness of f. □

Figure 2.2 illustrates the inclusion relationships we predict.

As can be seen, nearly-**RP** sets and nearly-**BPP** sets are related to the density of sets.

Definition 2.6.5 A set S is $\mathcal{C}$-$f(n)$-*close* if there exists a set B in $\mathcal{C}$ such that $\|(A \triangle B) \cap \Sigma^n\| \leq f(n)$ for almost all n.

Lemma 2.6.6 *Let f be a function on $\mathbb{N}$ such that $f(n) \in \omega(\log n)$. For $\mathcal{C} \in \{\mathbf{RP}, \mathbf{BPP}\}$, every $\mathcal{C}$-$2^{n-f(n)}$-close set is nearly-$\mathcal{C}$.*

Proof. We shall establish the lemma for $\mathcal{C} = \mathbf{BPP}$. Since $f(n) \in \omega(\log n)$, we have $\lim_{n\to\infty} \frac{c \cdot \log n}{f(n)} = 0$ for all number $c > 0$. This implies that $\lim_{n\to\infty} \frac{n^c}{2^{f(n)}} = 0$ for every constant $c > 0$. Hence, $2^{-f(n)} \in o\left(\frac{1}{p(n)}\right)$ for any polynomial p.

Let A be $\mathbf{BPP}$-$2^{n-f(n)}$-close. There exist a polynomial-time probabilistic Turing machine M and a set B such that $\|(A \triangle B) \cap \Sigma^n\| \leq 2^{n-f(n)}$ for almost all n. In other words, $\mathbf{Pr}_x[A(x) \neq B(x) \mid x \in \Sigma^n] \leq 2^{-f(n)}$.

We define $S = \{x \mid A(x) \neq B(x)\}$. Then, we have $\mathbf{Pr}_n[x \in S] \leq 2^{-f(n)}$. By the above calculation, for any polynomial p, $\mathbf{Pr}_n[x \in S] \leq \frac{1}{p(n)}$ for almost all n. This yields the desired consequence. □

2.7 Relevant Theories

In this section, we shall discuss several important and relevant branches of computational complexity theory.

2.7.1 Feasible Real Numbers

A real number is viewed as an infinite sequence of dyadic rational numbers which converges to it. If the convergence rate is fast enough and the nth element of the convergence sequence is effectively constructible, we can obtain a good approximation scheme for the real number in question. Ko and Friedman [55] called such real numbers *computable real numbers* and initiated *polynomial analysis* of the real numbers based on real numbers computable in polynomial time.

Definition 2.7.1 (Computable Real Numbers) [55] Let t be a function on $\mathbb{N}$. A real number r is *t-time computable* (*t-space computable*, resp.) if there exists a deterministic Turing machine M which, on input 1^n, produces dyadic rational number d_n in time $t(n)$ (using space $t(n)$, resp.) satisfying the condition $|r - d_n| \leq 2^{-n}$. We call the sequence $\{d_n\}_{n\in\mathbb{N}}$ a *convergence sequence* for r.

Definition 2.7.2 (Computable Sequence) A sequence $\{x_n\}_{n\in\mathbb{N}}$ of real numbers is *polynomial-time computable* (**P**-computable, for short) if there exists a polynomial p and a deterministic Turing machine M which, on input $(1^n, 1^k)$, produces a dyadic rational number d in time $p(n+k)$ satisfying $|d - x_n| \leq 2^{-k}$.

In polynomial analysis, functions map from $\mathbb{R}$ to $\mathbb{R}$. Although these real functions are intriguing, functions we are now interested only in functions that are "discrete" mappings from Σ^* to the unit interval $[0, 1]$. Here we modify the regular definition of feasible real functions to suit our setting.

Definition 2.7.3 (Computable Real-Valued Functions) A function f from Σ^* to $\mathbb{R}^+$ is called *t-time computable* (*t-space computable*, resp.) if there exists a deterministic Turing machine M which works in time

t (using space t, resp.) satisfying that $|M(x, 1^i) - f(x)| \leq 2^{-i}$ for all $i \in \mathbb{N}$ and all $x \in \Sigma^*$.

Lemma 2.7.4 *1. If x and y are* **P***-computable real numbers, then so are $-x$, $x + y$, $x \cdot y$, $\max\{x, y\}$, $\min\{x, y\}$, and $|x|$ (absolute value of x).*

2. Let x be a **P***-computable real number with $x \neq 0$. Then, the inverse $\frac{1}{x}$ is also a* **P***-computable real number.*

Proof. (1) Assume that $\{x_n\}_{n\in\mathbb{N}}$ and $\{y_n\}_{y\in\mathbb{N}}$ are convergence sequences witnessing x and y, respectively.

For $-x$, $\max\{x, y\}$, and $\min\{x, y\}$, consider the sequences $\{-x_n\}_{n\in\mathbb{N}}$, $\{\max\{x_n, y_n\}\}_{n\in\mathbb{N}}$, and $\{\min\{x_n, y_n\}\}_{n\in\mathbb{N}}$, respectively. For $x + y$, let $z_n = x_{n+1} + y_{n+1}$; then the sequence $\{z_n\}_{n\in\mathbb{N}}$ represents $x+y$. For $x \cdot y$, assume that the absolute values of x_n and y_n are bounded above by 2^k for some fixed natural number k. Then, we let $z_n = x_{n+k+1} \cdot y_{n+k+1}$. The sequence $\{z_n\}_{n\in\mathbb{N}}$ represents $x \cdot y$.

(2) Assume that $\{x_n\}_{n\in\mathbb{N}}$ is a convergence sequence witnessing x. Take an integer $k \in \mathbb{N}$ such that $|x| \geq 2^{-k+1}$. Therefore, $|x_n| = |x_n - x + x| \geq |x| - |x - x_n| \geq 2^{-k+1} - 2^{-n}$. So, for all $n \geq k$, $|x_n| \geq 2^{-k}$ holds. Let $x'_n = x_{2k+n+3}$. We have $|x'_n - x| < 2^{-2k-n-3}$. Let z_n be the inverse of x'_n rounded down to polynomially-many bits such that $|x'_n - \frac{1}{z_n}| < 2^{-2k-n-3}$. Then, we have

$$\left|x - \frac{1}{z_n}\right| \leq |x - x'_n| + \left|x'_n - \frac{1}{z_n}\right| < 2^{-2k-n-3} + 2^{-n} \leq 2^{-2k-n-2}.$$

Also we have

$$\left|\frac{1}{z_n}\right| = \left|\frac{1}{z_n} - x'_n + x'_n\right| \geq |x'_n| - \left|\frac{1}{z_n} - x'_n\right| \geq 2^{-k} - 2^{-2k-n-3} \geq 2^{-k-1}.$$

Hence,

$$\left|\frac{1}{x} - z_n\right| = \frac{|x - \frac{1}{z_n}|}{|x| \cdot |\frac{1}{z_n}|} \leq 2^k \cdot 2^{k+1} \cdot 2^{-2k-n-2} = 2^{-n}.$$

□

Lemma 2.7.5 *Let $\{a_n\}_{n\in\mathbb{N}}$ be an infinite convergence sequence in which each a_i is in $[0, 1]$. Let t be a function on $\mathbb{N}$. Let p be a time-constructible function on $\mathbb{N}$ and assume that p is increasing and unbounded. Also assume that $|a_{p(i)} - a_{p(j)}| \leq 2^{-i} + 2^{-j}$ for almost all $i, j \in \mathbb{N}$. If the sequence $\{a_n\}_{n\in\mathbb{N}}$ is t-time computable, then the limit $\lim_{n\to\infty} a_n$ is $O(n + t(p(n + 1) + n + 1))$-time computable. In particular, if $\{a_n\}_{n\in\mathbb{N}}$ is* **P***-computable and $p \in n^{O(1)}$, then $\lim_{n\to\infty} a_n$ is* **P***-computable.*

Proof. Since $\{a_n\}_{n\in\mathbb{N}}$ is t-time computable, there is a deterministic Turing machine M satisfying $|a_n - M(1^n, 0^i)| < 2^{-i}$ for all $n, i \in \mathbb{N}$. Let $N(1^n) = M(1^{p(n+1)}, 0^{n+1})$. More precisely, N is a deterministic Turing machine defined as below:

begin algorithm N
  input 1^n
  compute $n + 1$

compute $p(n+1)$ (needing $O(p(n+1)+n)$ steps)

$(*)$ simulate M on input $(1^{p(n+1)}, 0^{n+1})$

end.

If $p(n) < n$, then we use the convention that whenever M tries to check the i-th bit of the input, it writes down i in binary on a checking tape, and then it retrieves the answer automatically.

Then, the running time of N on input 1^n needs at most

$$c \cdot (\mathrm{Time}_M(1^{p(n+1)}, 0^{n+1}) + n) \leq c \cdot (t(p(n+1)+n+1)+n)$$

for almost all n, where c is an appropriate positive constant. We next claim that $|r - N(1^n)| \leq 2^{-n}$. This is seen as follows:

$$\begin{aligned} |r - N(1^n)| &= |c - M(1^{p(n+1)}, 0^{n+1})| \\ &\leq |c - a_{p(n+1)}| + |a_{p(n+1)} - M(1^{p(n+1)}, 0^{n+1})| \\ &\leq 2^{-n-1} + 2^{-n-1} = 2^{-n}. \end{aligned}$$

Therefore, r is computable in time $O(n + t(p(n+1)+n+1))$. □

The interested reader may refer to [54].

2.7.2 Kolmogorov Complexity

In this thesis, we use *time-bounded generalized Kolmogorov complexity* given by Hartmanis [38]. Intuitively, the Kolmogorov complexity of a finite string is the length of the shortest program that will generate the string.

To define Kolmogorov complexity, we need a notion of *universal Turing machines* that can simulate the behavior of any other Turing machines. We assume that $\{M_i\}_{i \in \mathbb{N}}$ is an effective enumeration of all Turing machines. Now consider the following machine U: on input $\langle i, x \rangle$, U simulates the ith machine M_i on input x.

A code is a prefix-code or instantaneous code if the set of code words is prefix-free, i.e., no code word is a prefix of another code word. A prefix-code is called *self-delimiting* if there is a Turing machine which (i) decides whether a given word is a code word and (ii) computes the decoding function.

Let us fix such a universal Turing machine U.

Definition 2.7.6 (Kolmogorov Complexity Sets) [38] A *time-bounded Kolmogorov complexity set* $\mathrm{KT}[g(n), t(n)]$ is the set of strings x such that, for some string y of length at most $g(|x|)$, U on input y outputs x in time $t(|x|)$. For a fixed string z, a *time-bounded conditional Kolmogorov complexity set relative to* z, $\mathrm{KT}[g(n), t(n)|z]$, is similarly defined but using $U(\langle y, z \rangle) = x$ instead of $U(y) = x$ in the above definition.

Lemma 2.7.7 *For any functions g, t and any string z,* $\mathrm{KT}[g(n), t(n)|z] \subseteq \mathrm{DTIME}(O(t(n) \cdot 2^{g(n)}))$.

Proof. For an output string x, we need to check at most all machines M_i coded by strings i of length $g(|x|)$, and, at each i, we simulate the machine M_i within $t(|x|)$ steps. □

For more discussion about Kolmogorov complexity, the reader may refer to [62].

2.7.3 Resource-Bounded Measure

In 1990, Lutz [63] developed *resource-bounded measure theory*. The following terms mainly follow [70].

A function d from Σ^* to $\mathbb{R}^+$ is a *martingale* if

$$d(w) = \frac{d(w0) + d(w1)}{2}$$

for all strings w. For every martingale d, we always have $d(w) \leq 2^{|w|} \cdot d(\lambda)$ for all w. Recall that s_i is the ith string in Σ^*. A martingale d *succeeds* on a set A if $\lim_{k\to\infty} \sup_{n\geq k} d(A[0...n]) = \infty$, where $A[0...n]$ denotes the string $x = x_0x_1\cdots x_n$ satisfying $x_i = \chi_A(s_i)$. Let $S^\infty[d]$ be a collection of all sets on which the martingale d succeeds.

The notion of *p-measure* captures in a way the "topological" size of a class.

Definition 2.7.8 (P-measure) cf. [64, 70] A complexity class $\mathcal{C}$ has *p-measure 0* if there exists a martingale d which is computable by a deterministic polynomial-time Turing machine such that $\mathcal{C} \subseteq S^\infty[d]$. A complexity class $\mathcal{C}$ has *p-measure 1* if the complement co-$\mathcal{C}$ has p-measure 0.

The class **E** has p-measure 1, but **P** has p-measure 0. We sometimes informally call a class *small* if it has p-measure 0. According to this terminology, the class **P** is small, while **E** is not small. One of the intriguing open question is whether **NP** is small. If **P** = **NP**, then obviously **NP** is small; hence, smallness of **NP** would follow from a collapse of **NP** down to **P**.

A *1-dimensional martingale system* (1-MS, for short) d is a function from $\mathbb{N} \times \Sigma^*$ to $\mathbb{R}^+$ such that d_k is a martingale for each $k \in \mathbb{N}$, where $d_k(w) = d(k, w)$. A set X is a *p-union of the p-measure 0 sets* $\{X_i\}_{i\in\mathbb{N}}$ if (i) $X = \bigcup_{j\in\mathbb{N}} X_j$, and (ii) there exists a polynomial-time computable 1-MS d such that $X_j \subseteq S^\infty[d_j]$ for every $j \in \mathbb{N}$.

***Lemma 2.7.9* [64]** *If X is a p-union of the p-measure 0 sets, then X has p-measure 0.*

We further say that $\mathcal{C}$ has *measure 0 in* **E** (*measure 1 in* **E**, resp.) if $\mathcal{C} \cap \mathbf{E}$ has p-measure 0 (p-measure 1, resp.). It is clear from the definition that if $\mathcal{C}$ has p-measure 0 and 1, then $\mathcal{C}$ has measure 0 and 1 in **E**, respectively. As an example, we note that, for a fixed positive constant c, the class DTIME($O(2^{c\cdot n})$) has measure 0 in **E** [70]. Another typical example of a class which has measure 0 in **E** is the collection of all p-m-complete sets for **E** (see [70]).

We shall give an important example of p-measure 0 sets: *immune sets* and *bi-immune sets*.

Definition 2.7.10 (Immune Sets and Bi-Immune Sets) Let $\mathcal{C}$ be any complexity class. A set S

is called $\mathcal{C}$-*immune* if S is infinite and S has no infinite subsets in $\mathcal{C}$. A set S is $\mathcal{C}$-*bi-immune* if S and its complement $\overline{S}$ are both $\mathcal{C}$-immune.

Proposition 2.7.11 **[69]** *Let $c > 0$. The class of all* DTIME$(O(2^{c \cdot n}))$*-bi-immune sets has p-measure* 1. *Hence, any class which contains no* DTIME$(O(2^{c \cdot n}))$*-bi-immune sets has p-measure* 0.

Proof. Let $\mathcal{C}$ be the class of non DTIME$(O(2^{c \cdot n}))$-bi-immune sets, and to obtain the desired result, we shall show that $\mathcal{C}$ has p-measure 0, because this obviously implies that the class of all DTIME$(O(2^{c \cdot n}))$-immune sets has p-measure 1.

Take a universal set A in $\mathbf{E}$ for DTIME$(O(2^{c \cdot n}))$, namely, DTIME$(O(2^{c \cdot n})) = \{A_i \mid i \in \mathbb{N}\}$, where $A_i = \{x \mid \langle x, i \rangle \in A\}$ for each i. We decompose $\mathcal{C}$ into infinitely-many subclasses $\{Y_m\}_{m \in \mathbb{N}}$ as follows:

$$Y_{2i-1} = \begin{cases} \{L \mid A_i \subseteq L\} & \text{if } \|A_i\| = \infty, \\ \emptyset & \text{otherwise;} \end{cases}$$

$$Y_{2i} = \begin{cases} \{L \mid A_i \subseteq \overline{L}\} & \text{if } \|A_i\| = \infty, \\ \emptyset & \text{otherwise.} \end{cases}$$

We define the 1-MS d as follows:

$$d(m, z) = \begin{cases} 1 & \text{if } z = \lambda, \\ 2d(m, w) & \text{if } s_{|w|} \in A_{\lfloor (m+1)/2 \rfloor}, \text{ and } b \not\equiv m \pmod 2, \\ 0 & \text{if } s_{|w|} \in A_{\lfloor (m+1)/2 \rfloor} \text{ and } b \equiv m \pmod 2, \\ d(m, w) & \text{otherwise,} \end{cases}$$

where w is the string satisfying $z = wb$ for some $b \in \{0, 1\}$.

To compute $d(m, z)$, we should check if $s_{|w|} \in A_{\lfloor (m+1)/2 \rfloor}$; however, since $s_{|w|}$ is of length $\mathrm{llog}(|w|)$ $(= \lfloor \log(|w| + 1) \rfloor)$, the computation time for checking if $s_{|w|} \in A_{\lfloor (m+1)/2 \rfloor}$ takes at most $c' \cdot 2^{c \cdot \mathrm{llog}(|w|)} \leq c' \cdot (|w| + 1)^c$, where c' is a constant. Thus, d is $\mathbf{P}$-computable.

It suffices to show by Lemma 2.7.9 that $\mathcal{C}$ is a p-union of the p-measure 0 sets $\{Y_m\}_{m \in \mathbb{N}}$. Let us see that $Y_m \subseteq S^\infty[d_m]$ for each $m \in \mathbb{N}$, where $d_m(w) = d(m, w)$. First consider $m \in \mathbb{N}$ such that m is odd. If A_m is finite, then we clearly have $Y_m \subseteq S^\infty[d_m]$. We then assume that A_m is infinite. Take any set $B \in Y_m$. Note that $A_{\lfloor (m+1)/2 \rfloor} \subseteq B$ since $B \in Y_m$. By our definition, we have $d_m(B[0..n]) = 2d_m(B[0..n-1])$ if $s_n \in A_{\lfloor (m+1)/2 \rfloor}$, and otherwise, $d_m(B[0..n]) = d_m(B[0..n-1])$. Since $\{n \mid s_n \in A_{\lfloor (m+1)/2 \rfloor}\}$ is infinite, $\limsup_{n \to \infty} d_m(B[0..n]) = \infty$. Thus, we have $B \in S^\infty[d_m]$. A similar argument works for the case where m is even. $\square$

Chapter 3

General Theory of Average Case Complexity

3.1 Introduction

Average-case analyses have been performed to measure the complexity of algorithms and to obtain a better understanding of the behavior of algorithms when input instances are given with some probability. In this type of analysis, we should take into consideration that instances of a particular algorithm occur with some probability. In contrast to the approach of from worst-case analysis, a "problem" here is a pair consisting of a *set of instances* and an *input distribution* which designates the probability of each instance. These problems are called *distributional problems*, *randomized problems*, or *random problem*.

Classical average-case analysis uses the expected running time or tape squares in use over all input instances of the same length. Although the notion is simple and intuitive, it is not a basis for a consistent and coherent theory of average-case complexity. Levin's theory of average **NP**-completeness uses instead the notion of *polynomial on the average* and also *polynomial-time many-one reducibility* among distributional decision problems with crucial conditions, the so-called *domination conditions* for reduction functions. This constraint is essential in the theory to make the reducibility transitive and to make the average polynomial-time computable class closed under the reductions. In this chapter, we shall review Levin's theory and further cultivate a general framework of average-case complexity theory; we shall focus mainly on the notions of *polynomial on μ-average* and *polynomial domination relations* on which "domination conditions" rely.

Levin's notion of functions being *polynomial on μ-average* has been expanded into random functions to cope with the average-case analysis of randomized algorithms.

Section 3.2 will begin with the formal definition of *distributions* (or distribution functions) and *(probability) density functions* (or probability distributions). For practical reasons, we shall introduce *semi-distributions* by eliminating the condition that distributions converge to 1 as input strings get larger. Of particular importance is the *standard distribution* that assigns to each string the probability of the string

being chosen at random.

In Section 3.3, we shall introduce the notion of *t on μ-average* which generalizes Levin's original notion of "polynomial on μ-average." Briefly, a function g is called *polynomial on μ-average* if, for some positive number k, the expected value of $|x|^{-1} \cdot g(x)^{1/k}$ over *all* input strings x under distribution μ converges. The reader may notice that the functions which are polynomial on μ-average are in general not time-constructible, or even computable.

Section 3.4 will introduce of the notions of *domination relations* and *equivalence relations* among distributions. These relations are the essential ingredients of Levin's domination conditions for average-case reductions.

Section 3.5 will introduce two types of average-case complexity classes, "randomized" version of worst-case complexity classes and "average time/space bounded" version of worst-case complexity classes to describe the classes which consist of distributional decision problems. We shall introduce two types of notations: $\mathrm{Dist}(\mathcal{C}, \mathcal{F})$ for the randomized complexity classes, and $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ for average complexity classes. From the algorithmic point of view, we can consider sets of distributional decision problems (D, μ) whose underlying problem D is solved in time polynomial on μ-average. In particular, we shall introduce average-case analogues of important complexity classes, such as **P**, **NP**, **BPP**, and **PSPACE**. However, the reader should note that our average running time and tape space are not measured by time-constructible functions, and as a result, fundamental relationships among the average complexity classes are essentially different from those among worst-case complexity classes.

Major Contributions. This chapter extends Schapire's characterization of "polynomial on μ-average" to a more general notion of "t on μ-average." The reader who is familiar with Levin's original definition may find it interesting that the arguments used in this thesis to show that a given function is polynomial on μ-average are very different from those used in the literature. The basis of this chapter comes from Schuler and Yamakami [97].

Lemma 3.3.15 gives a simple but sufficient condition for proving a given function g to be polynomial on μ-average. The lemma actually shows that if $g(x) \leq c \cdot n^k + (c/n^2 \hat{\mu}(A_i^n))^k$ for all i with $1 \leq i \leq n$ and for all $x \in A_i^n$, then g is polynomial on μ-average, where $\{A_i^n\}_{1 \leq i \leq n}$ is a partition of Σ^n.

Lemma 3.4.16 shows that, for a random function h with its random-input domain Γ, if $\mu \preceq_\Gamma^{\mathrm{avrp}} \nu$ and $\hat{\nu}(\{(x,s) \mid h(x,s) > q(r \cdot |x|)\}) < 1/r$, then h becomes polynomial on μ-average.

A generalization of Schapire's result is presented as in Proposition 3.3.17. The proposition actually shows that, for a good set $\mathcal{T}$ of strictly increasing functions, g is $\mathcal{T}$ on μ-average if and only if the expectation of the values $|x|^{-1} \cdot t^{-1}(g(x))$, over all input strings x, converges for some function t in $\mathcal{T}$.

Lemmas 3.3.20 and 3.3.21 are new results. Lemma 3.3.20 shows a sufficient condition for a random function g to be polynomial on μ-average, while Lemma 3.3.21 gives a necessary condition for g.

Lemma 3.3.22 shows that, provided a function h is polynomial on μ-average and a random function g is polynomial on $\mu \times \eta$-average, if $h(f(x)) \cdot \hat{\eta}(f(x)) \geq 1$ for all x and $\lambda x.|f(x)|$ is polynomial on μ-average, then the composition $\lambda x s.g(x, h(x), s)$ is polynomial on μ-average.

Proposition 3.5.20 shows that, for example, Aver($\mathbf{BPP}, \mathcal{F}$) is weakly **PP**-descriptive.

Lemma 3.5.21 shows that Aver($\mathbf{P}, \mathcal{F}$) and Aver($\mathbf{PSPACE}, \mathcal{F}$) are closed under weak description, and this lemma combined with Proposition 3.5.22 leads to Corollary 3.5.23. The corollary shows that $\mathbf{P} \neq \mathbf{BPP}$ implies the separation of Aver($\mathbf{BPP}, *$) from Aver($\mathbf{P}, *$), whereas the two average-case complexity classes collapse if $\mathbf{P} = \mathbf{PP}$. (These results follow Proposition 3.5.22, which shows an extensive generalization of a result given by Karg and Schuler [48].)

Our notion of nondeterministic average polynomial-time is clearly distinct from what has been discussed elsewhere. In particular, whereas $\mathbf{P} =?\mathbf{NP}$ is a lon-standing open question, Theorem 3.5.24 makes the interesting observations that Aver($\mathbf{P}, *$) $\neq$ Aver($\mathbf{NP}, *$).

The new characterizations of Aver($\mathbf{NP}, \mathcal{F}$) and Aver($\mathbf{BPP}, \mathcal{F}$) are presented in Proposition 3.5.30 and Proposition 3.5.33. In particular, the Amplification Lemma (Lemma 3.5.31) in our average-case setting, followed by Proposition 3.5.33, is fundamental and finds many applications in later chapters. The Amplification Lemma amplifies the success probability of randomized algorithms which make bounded-errors.

3.2 Distributions and Density Functions

Average-case complexity theory handles problems whose input instances occur with specified probabilities. This section will introduce the basic concepts of *distribution* (or *distribution functions*) and *(probability) density functions*. Using the terminology in Section 2.2, our sample space Ω consists of all finite strings over $\{0, 1\}$, and the σ-field is $(\Omega, \mathbb{F})$, where $\mathbb{F}$ is the power set of Ω. We consider a discrete probability measure on $(\Omega, \mathbb{F})$ and call it a (probability) density function. Intuitively, a density function provides a probability that instance x occurs. A distribution, on the other hand, indicates the total probability over all instances smaller than or equal to a give instance. For practical reasons, we also use the notion of *semi-distributions*.

Definition 3.2.1 (Distributions) A *semi-distribution* (or *semi-distribution function*) μ is an increasing function from Σ^* to the unit real interval $[0, 1]$. A *distribution* (or *distribution function*) is a semi-distribution which converges to 1, i.e., $\lim_{x \to \infty} \mu(x) = 1$.

We do not avoid the possibility that the semi-distribution μ always takes the value 0, i.e., $\hat{\mu}(x) = 0$ for all x; we call such μ *trivial*. We remark that there is no feasible way to determine in general whether a given semi-distribution μ is trivial. Nevertheless, we are primarily interested in non-trivial semi-distributions.

Ben-David, Chor, Goldreich, and Luby [9], among others, often use semi-distributions in their arguments instead of *full* distributions because their semi-distributions can be normalized to *full* distributions without changing the complexity. However, this normalization is not always possible. See Section 4.2 for more discussion.

Next we shall define (probability) density functions which are probability measures on the σ-field $(\Sigma^*, \mathcal{P}(\Sigma^*))$.

Definition 3.2.2 (Density Functions) For a distribution μ, its associated *(probability) density function* $\hat{\mu}$ is defined by the *probability* $\hat{\mu}(x)$ on input string x as follows:

$$\hat{\mu}(x) = \begin{cases} \mu(\lambda) & \text{if } x = \lambda, \\ \mu(x) - \mu(x^-) & \text{otherwise.} \end{cases}$$

(Note that a density function is often called a *probability distribution* in much of the literature (see, e.g., [11]) and should not be confused with a "distribution.")

The reader must keep in mind that if μ is a distribution, then $\hat{\mu}(\lambda) = \mu(\lambda)$ by our definition. Also note that $\mu(x) = \sum_{z:z\leq x} \hat{\mu}(z)$ holds for all strings x.

We have already seen the notation $\mathbf{Pr}[\mathcal{E}]$ for event $\mathcal{E}$ based on a sample space Ω which consists of finite or infinite sequences over $\{0,1\}$. We reserve this notation for events where each bit of a sequence is chosen *at random*; that is, for any property Q on Σ^* and any subset E of Ω,

$$\mathbf{Pr}_s[Q(s) \mid s \in E] = \sum_{s \in E} 2^{-|s|} \cdot [Q(s)].$$

For convenience, we use the notation "$\hat{\mu}(x) \propto g(x)$" to mean that the probability $\hat{\mu}(x)$ on input string x is proportional to the value $g(x)$ for every x; more precisely, there exists a constant $c > 0$ such that $\hat{\mu}(x) = c \cdot g(x)$ for all x. This c is called the *normalizing constant* for g.

For a distribution μ and a set S, let $\hat{\mu}(S)$ denote the sum $\sum_{x\in S} \hat{\mu}(x)$. For example, $\hat{\mu}(\Sigma^{\leq n}) = \mu(1^n)$ and $\hat{\mu}(\Sigma^*) = \lim_{x\to\infty} \mu(x)$.

Recall that distributions are mappings from the infinite set Σ^* to the unit real interval $[0,1]$. We also cope with *ensembles of finite input distributions* instead of "infinite" distributions. Given a distribution μ and a natural number n, the *conditional distribution* of μ on Σ^n, denoted by μ_n, is the function from Σ^n to $[0,1]$ that is defined by its density function $\hat{\mu}_n$ as follows: for each $x \in \Sigma^n$,

$$\hat{\mu}_n(x) = \frac{\hat{\mu}(x)}{\hat{\mu}(\Sigma^n)}$$

whenever $\hat{\mu}(\Sigma^n) \neq 0$; otherwise, $\hat{\mu}_n(x)$ is undefined. Similarly, let $\mu_{\leq n}$ be the *conditional distribution* of μ on $\Sigma^{\leq n}$ that is defined by its density function $\hat{\mu}_{\leq n}$ as follows: for each x in $\Sigma^{\leq n}$,

$$\hat{\mu}_{\leq n}(x) = \frac{\hat{\mu}(x)}{\hat{\mu}(\Sigma^{\leq n})}$$

if $\hat{\mu}(\Sigma^{\leq n}) \neq 0$; otherwise, let $\mu_{\leq n}$ be undefined. In general, it holds that $\hat{\mu}(x) \leq \hat{\mu}_{\leq n}(x)$ for all $x \in \Sigma^{\leq n}$ and $\hat{\mu}_{\leq n}(x) \leq \hat{\mu}_n(x)$ for all $x \in \Sigma^n$ if μ_n and $\mu_{\leq n}$ are both defined.

Similarly, we denote by $\mu_{\geq n}$ the conditional distribution of μ on $\Sigma^{\geq n}$.

For a nonempty finite domain D, the *uniform distribution* on D is defined to have the probability $1/\|D\|$ for every x in D; that is, $\hat{\mu}(x) = \mathbf{Pr}_s[x = s \mid s \in D] = \frac{1}{\|D\|}$. For example, μ_n is the uniform distribution on Σ^n if $\hat{\mu}(x) = 2^{-|x|}$ for all $x \in \Sigma^n$.

For a set S, μ is called *positive on* S if $\hat{\mu}(x) > 0$ for all $x \in S$; in particular, if $S = \Sigma^*$, then we say that μ is *positive*.

Another important class of distributions, called *flat distributions*, was introduced by Gurevich [36].

Definition 3.2.3 (Flat Distributions) [36] A distribution μ is called *flat* if there exists a real number $\epsilon > 0$ such that $\hat{\mu}(x) \leq 2^{-|x|^{\epsilon}}$ holds for almost all x. Notationally, FLAT denotes the collection of all flat distributions.

For a function f on Σ^*, we write $\mu_{f^{-1}}$ to denote the distribution defined by its probability $\hat{\mu}_{f^{-1}}(x) = \hat{\mu}(\{z \mid f(z) = x\})$.

Recall that distributions defined in Definition 3.2.1 are unary functions. We can also consider *multi-dimensional distributions*. For a k-dimensional vector $(x_1, x_2, \ldots, x_k)$ over Σ^*, if a density function $\hat{\mu}$ is defined, then let

$$\mu(x_1, x_2, \ldots, x_k) = \hat{\mu}(\{(z_1, z_2, \ldots, z_k) \mid z_1 \leq x_1, z_2 \leq x_2, \ldots, z_k \leq x_k\}).$$

Using an effective encoding of k-dimensional vectors into strings (discussed in Section 2.5), however, we can always identify probability $\hat{\mu}(x_1, x_2, \ldots, x_k)$ with probability $\hat{\mu}(\langle x_1, x_2, \ldots, x_k \rangle)$ defined on Σ^*. In this sense, we do not need to consider all multi-dimensional distributions.

To simplify the descriptions of distributions, we use $\hat{\mu}(x, y)$ and $\hat{\mu}(x, y, z)$ to denote $\hat{\mu}(\langle x, y \rangle)$ and $\hat{\mu}(\langle x, y, z \rangle)$, respectively. We also use a simplified definition of distributions. For example, the following schematic definition "$\hat{\mu}(s_i, x, 1^n) = \hat{\nu}(s_i) \cdot \hat{\eta}(x) \cdot 2^{-2\mathrm{llog}(n)-1}$" really means that

$$\hat{\mu}(u) = \begin{cases} \hat{\nu}(s_i) \cdot \hat{\eta}(x) \cdot 2^{-2\mathrm{llog}(n)-1} & \text{if } u = \langle s_i, x, 1^n \rangle \text{ for some } i, x, n, \\ 0 & \text{otherwise.} \end{cases}$$

One of the most important distributions is the positive distribution where each string is chosen "uniformly" at random. This distribution is called *standard*. In this thesis, we use the standard distribution ν_{stand}, whose values are dyadic rational numbers, that are easily sampled by the following randomized algorithm: pick a natural number n randomly and then pick a string of length n randomly. To pick a natural number "uniformly" at random, we first define the translation Tr by $Tr(\lambda) = \lambda$, $Tr(00s) = 0Tr(s)$, $Tr(11s) = 1Tr(s)$, and $Tr(01s) = Tr(10s) = \#$ for a string s, where $\#$ is the terminal symbol different from 0 and 1. We then generate a string of the form $s01$ or $s10$ such that $Tr(s)$ is the nth string with respect to the standard order in Σ^*. By a simple estimation, we have $\hat{\nu}_{\mathrm{stand}}(x) = 2^{-|x|-2\mathrm{llog}(|x|)-1}$, where $\mathrm{llog}(n) = \lfloor \log(n+1) \rfloor$.

Definition 3.2.4 (Standard Distribution on Σ^*) The *standard distribution on* Σ^* is denoted by $\hat{\nu}_{\mathrm{stand}}$ and defined as $\hat{\nu}_{\mathrm{stand}}(x) = 2^{-|x|-2\mathrm{llog}(|x|)-1}$.

From this definition, it follows that, for every x,

$$\frac{1}{8(|x|+1)^2 \cdot 2^{|x|}} \leq \hat{\nu}_{\mathrm{stand}}(x) \leq \frac{1}{2(|x|+1)^2 \cdot 2^{|x|}}$$

since $\log(n+1)-1 \le \mathrm{llog}(n) \le \log(n+1)$. Moreover, $\nu_{\mathrm{stand}}(1^n) = 1 - 3\cdot 2^{-\mathrm{llog}(n)-1} + (n+2)\cdot 2^{-2\mathrm{llog}(n)-1}$. This is seen as follows:

$$\begin{aligned}
\nu_{\mathrm{stand}}(1^n) &= \sum_{i=0}^{n} 2^{-2\mathrm{llog}(i)-i-1}\cdot 2^i \;=\; \frac{1}{2}\cdot\left(\sum_{i=0}^{2^{\mathrm{llog}(n)-1}} 2^{2\mathrm{llog}(i)} + \sum_{i=2^{\mathrm{llog}(n)}}^{n} 2^{2\mathrm{llog}(i)}\right)\\
&= \frac{1}{2}\cdot\left(1-\frac{1}{2^{\mathrm{llog}(n)}}\right) + \frac{1}{2}\cdot\left(\frac{n+2}{2^{2\mathrm{llog}(n)+1}} - \frac{1}{2^{\mathrm{llog}(n)}}\right)\\
&= 1 - \frac{3}{2^{\mathrm{llog}(n)+1}} + \frac{n+2}{2^{2\mathrm{llog}(n)+1}}.
\end{aligned}$$

Lemma 3.2.5 *For every string x,*

$$\nu_{\mathrm{stand}}(x) = 1 - \frac{3}{2^{\mathrm{llog}(|x|-1)+1}} + \frac{|x|+1}{2^{2\mathrm{llog}(|x|-1)+1}} + \frac{k+1}{2^{2\mathrm{llog}(|x|)+|x|+1}},$$

where x is the kth string of $\Sigma^{|x|}$. (N.B. $0^{|x|}$ is the 0th string of $\Sigma^{|x|}$.)

Proof. Let $n = |x|$. Notice that $\nu_{\mathrm{stand}}(x) = \nu_{\mathrm{stand}}(1^{n-1}) + \sum_{i=0}^{k} \hat{\nu}_{\mathrm{stand}}(s_i^n)$. The first term is equal to $1 - 3\cdot 2^{-\mathrm{llog}(n-1)+1} + (n+1)\cdot 2^{-2\mathrm{llog}(n-1)+1}$. The last term is equal to $(k+1)\cdot\hat{\nu}_{\mathrm{stand}}(1^n)$, and thus $(k+1)\cdot 2^{-2\mathrm{llog}(n)-n-1}$. □

In particular, $\nu_{\mathrm{stand}}(1^{p(i)}) = 1 - 2^{-i}$, where $p(i) = 2^{i+1}-2$, and thus we get

$$|\nu_{\mathrm{stand}}(1^{p(i)}) - \nu_{\mathrm{stand}}(1^{p(j)})| < 2^{-i} + 2^{-j}$$

for all $i, j \in \mathbb{N}$. This inequality will be used later.

As the reader can see, we can generalize ν_{stand} to the default distribution ν_{stand}^k defined on the set of k-tuples, $\{\langle x_1, x_2, \ldots, x_k\rangle \mid x_1, x_2, \ldots, x_k \in \Sigma^*\}$ by

$$\hat{\nu}_{\mathrm{stand}}^k(x_1, x_2, \ldots, x_k) = \hat{\nu}_{\mathrm{stand}}(x_1)\cdot\hat{\nu}_{\mathrm{stand}}(x_2)\cdots\hat{\nu}_{\mathrm{stand}}(x_k).$$

In some papers, the standard distribution is defined as $\hat{\nu}_{\mathrm{stand}}(x) = \frac{6}{\pi^2}\cdot(|x|+1)^{-2}\cdot 2^{-|x|}$ or $\hat{\nu}_{\mathrm{stand}}(x) = 2^{-|x|}/(|x|+1)(|x|+2)$; these definitions are not essentially different from the one used here. Levin [60] uses $|x|^{-2}\cdot 2^{-|x|}$ for $\hat{\nu}_{\mathrm{stand}}(x)$ for notational convenience (with the normalizing constant $\frac{6}{\pi^2}$).

Although the standard distribution is sometimes called "uniform" (e.g., in [9]), actually only its conditional distribution is uniform for all lengths n. We note that there are other ways to define a "standard" distribution (see Gurevich [36] for more discussion).

We also use the standard distribution ν_{tally} that takes positive values only on $\{0\}^*$ (or sometimes $\{1\}^*$).

Definition 3.2.6 (Standard Distribution on $\{0\}^*$) The *standard distribution on* $\{0\}^*$ is denoted by ν_{tally} and is defined as

$$\hat{\nu}_{\mathrm{tally}}(x) = \begin{cases} 1/2 & \text{if } x = \lambda,\\ 2^{-2\mathrm{llog}(n)-1} & \text{if } x = 0^n \text{ for some } n > 0,\\ 0 & \text{otherwise.}\end{cases}$$

For the sake of convenience, we sometimes use the same notation ν_{tally} to mean the standard distribution on $\{1\}^*$.

Definition 3.2.7 (Default Distribution) For a random-input domain Γ and a semi-distribution μ, we define the default semi-distribution μ_Γ induced from μ and Γ as

$$\hat{\mu}_\Gamma(x,s) = \begin{cases} U_\Gamma(x) \cdot \hat{\mu}(x) \cdot 2^{-|s|} & \text{if } s \in \Gamma(x), \\ 0 & \text{otherwise,} \end{cases}$$

where U_Γ is the rarity function of Γ.

In the special case that Γ is almost total, for every subset $S \subseteq \Gamma$, it holds that

$$\hat{\mu}_\Gamma(S) = \sum_x \hat{\mu}(x) \cdot \mathbf{Pr}_s[(x,s) \in S \mid s \in \Gamma(x)].$$

A function g from Σ^* to $\mathbb{R}^\infty$ is called *degenerative under* μ if $\hat{\mu}(\{x \mid g(x) = \infty\}) = 0$.

A set $\mathcal{F}$ of distributions is *closed under k-addition* if, for any k semi-distributions, $\mu_1, \mu_2, \ldots, \mu_k$, from $\mathcal{F}$, the distribution ν defined by $\hat{\nu}(x) = \sum_{i=1}^k \frac{1}{k} \cdot \hat{\mu}_i(x)$ belongs to $\mathcal{F}$.

For two distributions μ and ν, let $\mu \oplus \nu$ denote the distribution η such that $\hat{\eta}(x) = \frac{1}{2}\hat{\mu}(u)$ if $x = 0u$ for some u; $\hat{\eta}(x) = \frac{1}{2}\hat{\nu}(u)$ if $x = 1u$ for some u; and $\hat{\eta}(x) = 0$, otherwise. We say that a set $\mathcal{F}$ of distributions is *closed under* $\oplus$ if, for any two distributions μ and ν in $\mathcal{F}$, $\mu \oplus \nu$ is also in $\mathcal{F}$.

3.3 A Notion of Easy-on-Average

To establish a consistent and coherent theory of average-case complexity theory, first we must examine a fundamental notion of computational "average complexity" of algorithms, after which we will look at Levin's innovative idea of how to amend the naive definition.

3.3.1 Naive Definition of Average Polynomial Time

In worst-case complexity theory, a problem has *$t(n)$-time complexity* if there is an algorithm M computing the problem which satisfies the inequality $\mathrm{Time}_M(x) \leq t(n)$ for almost all natural numbers n and all inputs $x \in \Sigma^n$. A naive notion of average-case complexity, however, is given by the expected running time (or space) of an algorithm over all instances of the same length under a certain conditional distribution. More precisely, an algorithm M which works in time t requires the inequality that $\sum_{x:|x|=n} \mathrm{Time}_M(x) \cdot \hat{\mu}_n(x) \leq t(n)$ for almost all n. This natural formulation of an average-case complexity might seem to be a start on a general theory of average-case complexity. Unfortunately, this definition has several deficiencies. We will see several examples below. For brevity, we say that a function g "expected polynomial on μ-average" if there exists a constant $k > 0$ such that, for all $n \in \mathbb{N}$, $\sum_{x:|x|=n} g(x) \cdot \hat{\mu}_n(x) \leq n^k + k$ [108].

The first example is related to the multiplication of functions which are expected polynomial on μ-average.

Example 3.3.1 **[9]** Consider a function g which is defined on strings of length n as follows: let $g(x)$ be 2^n on n inputs x, but let $g(x)$ be n on the other inputs. Now take a conditional distribution μ_n: $\hat{\mu}_n(x) = 2^{-n}$ for all x of length n. It is easy to see that the expectation of g, $\sum_{x:|x|=n} g(x) \cdot \hat{\mu}_n(x)$, is at most $2n$, but the expectation of g^2, $\sum_{x:|x|=n} (g(x))^2 \cdot \hat{\mu}_n(x)$, exceeds $n \cdot 2^n$.

Even if g is expected polynomial on μ-average, g^2 is no longer expected polynomial on μ-average.

The second example deals with the composition of functions which are expected polynomial on μ-average.

Example 3.3.2 **[9]** For simplicity, let n be of the form $4m$ for some integer $m > 0$. Consider the following machine M: M is computed in $2^{n/2}$ steps on $2^{n/2}$ strings of length n and outputs a string of length $\sqrt{2^{n/2}}$, and, for the rest of the inputs, M requires n^2 steps to output a string of length n. The expected running time of this machine M is at most $1 + n^2 \cdot 2^{-n/2}$, and hence, M runs in expected polynomial time on μ-average. Now consider another machine N which needs n^3 steps. Clearly N runs in polynomial time. However, the composition of two machines, $N(M(x))$, needs $2^{n/4} + n^3 \cdot 2^{-n/2}$ steps on the average.

Hence, this naive notion of "expected polynomial on the average" is not closed under composition.

Example 3.3.3 **[9]** Consider a problem on directed graph $G = G(V, E)$, where V is a set of vertices and E is a set of edges. Let $\|V\| = n$ and $\|E\| = m$. Assume that there is an algorithm M which works on this graph G in time $t(G)$, where $t(G) = 2^n$ if $m < n^{3/2}$; otherwise, $t(G) = n^2$. Suppose the graph G is given by its (incident) matrix representation. Since encodings of graphs are presented by n vertices, the average is taken over all graphs G with n nodes. The expected running time $\sum_G t(G) \cdot \hat{\mu}_n(G)$ is at most

$$\sum_{G:m<n^{3/2}} 2^{-\binom{n}{2}} \cdot 2^n + \sum_{G:m\geq n^{3/2}} 2^{-\binom{n}{2}} \cdot n^2 < 1 + n^2.$$

On the other hand, suppose the graph is given by its adjacency list. Then, the expected running time is not expected polynomial on μ-average.

The naive definition is dependent on the particular encoding of instances of a given problem.

3.3.2 Levin's Definition

As we have seen, the naive definition is not suitable for a coherent theory. In 1984, Levin [60] instead proposed a new measure of "average polynomial-time." To understand his measure, we again take a close look at the definition of worst-case complexity measure. Recall that an algorithm M needs polynomial time if $\mathrm{Time}_M(x) \leq |x|^k$ for almost all x. Here we transform this inequality into another form of inequality: $|x|^{-1} \cdot \mathrm{Time}_M(x)^{1/k} \leq 1$. A natural idea then is that the expectation of $|x|^{-1} \cdot \mathrm{Time}_M(x)^{1/k}$ over all strings of length n, is bounded by 1; namely, $\sum_{x:|x|=n} |x|^{-1} \cdot \mathrm{Time}_M(x)^{1/k} \cdot \hat{\mu}_n(x) \leq 1$. Levin was motivated by this inequality, but he went one step further and took this average over *all finite* strings. He defined his average-case complexity measure "polynomial on μ-average" by requiring that $\sum_{x:x\neq\lambda} |x|^{-1} \cdot \mathrm{Time}(x)^{1/k} \cdot \hat{\mu}(x) < \infty$.

Since his first paper appeared in 1984 at the 16th STOC conference, several criticisms of Levin's complexity measure have arisen. One of them is that his formulation does not seem to reflect the polynomiality of the running time on the average. In 1990, E. Schapire [88] wrote a technical report in which he exhibited an emerging theory of Levin's average-case **NP**-completeness and gave an interesting insight into Levin's central notion of "polynomial on μ-average." He gave an equivalent formulation of this notion.

This thesis modifies Schapire's characterization of *polynomial on μ-average* and introduces a more general notion of *t on μ-average* for an arbitrary function t.

Definition 3.3.4 (t on μ-Average) [88] Let t be a function on $\mathbb{R}^+$ and let μ be a distribution. Let g be a function from Σ^* to $\mathbb{R}^{+\infty}$.

1. The function g is *t on μ-average* if $\hat{\mu}(\{x \mid g(x) > t(|x| \cdot r)\}) < 1/r$ for any positive real number r.

2. The function g is *$\mathcal{T}$ on μ-average* if there exists a function $t \in \mathcal{T}$ such that g is t on μ-average.

Schapire actually used a function with two variables, $t(n, r)$, instead of the form $t(n \cdot r)$ in the above definition. In most cases, however, there is no practical difference between these two functions. For this reason, we use the above definition throughout this thesis. (This issue has been thoroughly studied by Karg and Schuler [48], and the interested reader may refer to it.)

Notice, from Definition 3.3.4, that if g is t on μ-average, then g is degenerative under μ. Moreover, it immediately follows from Definition 3.3.4 that increasing the value of r also increases the probability weight of the set of strings x with the property that $g(x) \leq t(|x| \cdot r)$, which is $1 - 1/r$.

Definition 3.3.5 (Polynomial/Logarithmic on μ-Average) A function g from Σ^* to $\mathbb{R}^{+\infty}$ is *polynomial on μ-average* if there exists a polynomial p such that g is p on μ-average. Similarly, g is *logarithmic on μ-average* if g is q on μ-average for some logarithmic function q (i.e., $q(z) = c \log z + d$ for some constants $c, d \in \mathbb{R}$).

The notion of *polynomial on μ-average* was first introduced in [60] and used in [30, 88, 36, 9, 111], while the notion of *logarithmic on μ-average* was defined by [9] and also used in [32].

Lemma 3.3.6 *Let μ be a distribution. For any function g from Σ^* to $\mathbb{R}^{+\infty}$, if g is logarithmic on μ-average, then the function $\lambda x.2^{g(x)}$ is polynomial on μ-average.*

Proof. Let us assume that g is q on μ-average for some logarithmic function q. Suppose without loss of generality that $q(z) = c \log z + d$ for constants $c, d > 0$. Notice that q is strictly increasing. Using this fact, we have

$$\begin{aligned} \hat{\mu}(\{x \mid g(x) > c \log(|x| \cdot r) + d\}) &= \hat{\mu}(\{x \mid 2^{g(x)} > 2^{c \log(|x| \cdot r) + d}\}) \\ &= \hat{\mu}(\{x \mid 2^{g(x)} > 2^d \cdot (|x| \cdot r)^c\}). \end{aligned}$$

Since g is q on μ-average, we obtain $\hat{\mu}(\{x \mid 2^{g(x)} > 2^d \cdot (|x| \cdot r)^c\}) < 1/r$. Since the function $\lambda z.(2^d z^c)$ is a polynomial, $\lambda x.2^{g(x)}$ becomes polynomial on μ-average. □

Our definition measures the probability of the event in which $g(x)$ exceeds $p(|x| \cdot r)$ over *all* strings x under distribution μ. Instead of this distribution, we can consider two ensembles of conditional distributions, $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ and $\{\mu_{\geq n}\}_{n\in\mathbb{N}}$.

Definition 3.3.7 **(t on Average w.r.t an Input Ensemble) cf. [43]** Let g be a function from Σ^* to $\mathbb{R}^{+\infty}$ and let t be a function on $\mathbb{R}^+$. We say that g *is t on average with respect to* $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ if $\hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid g(x) > t(|x| \cdot r)\}) < 1/r$ holds for all real numbers $r > 0$ for which $\mu_{\leq n}$ is defined. Similarly, we define the notion that g *is t on average with respect to* $\{\mu_{\geq n}\}_{n\in\mathbb{N}}$ by replacing $\mu_{\leq n}$ with $\mu_{\geq n}$.

Lemma 3.3.8 *Let μ be a distribution, g a function from Σ to $\mathbb{R}^{+\infty}$, and t a function on $\mathbb{R}^+$.*

1. *If g is t on μ-average, then g is $\lambda z.t(dz)$ on average w.r.t. $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ for some positive integer d.*
2. *If g is t on average with respect to $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$, then g is $\lambda z.t(2z)$ on μ-average.*

Proof. Let μ be a distribution. We can assume without loss of generality that $g(x) < \infty$ for all x. Take the minimal integer $k_0 \in \mathbb{N}$ such that $\hat{\mu}(\Sigma^{\leq k_0}) > 0$. This k_0 is the minimal number for which $\mu_{\leq k_0}$ is defined.

(1) First notice that $\hat{\mu}(\Sigma^{\leq n})$ equals $\mu(1^n)$, and thus $\lambda n.\hat{\mu}(\Sigma^{\leq n})$ is increasing. (This is not true for $\lambda n.\hat{\mu}(\Sigma^n)$.) Then, define $d = \min\{d' \in \mathbb{N} \mid \hat{\mu}(\Sigma^{\leq k_0}) \geq \frac{1}{d'}\}$. Note that, for any set B, if $\mu_{\leq n}$ is defined, then we have

$$\hat{\mu}_{\leq n}(B^{\leq n}) = \frac{\hat{\mu}(B^{\leq n})}{\hat{\mu}(\Sigma^{\leq n})} \leq \frac{\hat{\mu}(B)}{\hat{\mu}(\Sigma^{\leq k_0})} \leq d \cdot \hat{\mu}(B).$$

Now assume that g is t on μ-average. This assumption implies that, for any integer $n \geq k_0$,

$$\hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid g(x) > t(d \cdot |x| \cdot r)\}) \leq d \cdot \hat{\mu}(\{x \mid g(x) > t(|x| \cdot dr)\}) < d \cdot \frac{1}{dr} = \frac{1}{r}.$$

Therefore, g is $\lambda z.t(dz)$ on average w.r.t. $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$.

(2) Assume that g is t on average with respect to $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$; that is, $\hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid g(x) > t(|x| \cdot r)\}) < 1/r$ for all $r > 0$. We fix $r > 0$ arbitrarily. From the fact that $\hat{\mu}(B) = \hat{\mu}(\Sigma^{\leq n}) \cdot \hat{\mu}_{\leq n}(B)$ for any set $B \subseteq \Sigma^{\leq n}$, it immediately follows that, for $n \geq k_0$,

$$\hat{\mu}(\{x \in \Sigma^{\leq n} \mid g(x) > t(|x| \cdot 2r)\}) = \hat{\mu}(\Sigma^{\leq n}) \cdot \hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid g(x) > t(|x| \cdot 2r)\}) < \frac{\hat{\mu}(\Sigma^{\leq n})}{2r}.$$

Taking the limit, it follows that

$$\hat{\mu}(\{x \mid g(x) > t(|x| \cdot 2r)\}) = \lim_{n\to\infty} \hat{\mu}(\{x \in \Sigma^{\leq n} \mid g(x) > t(|x| \cdot 2r)\}) \leq \lim_{n\to\infty} \frac{\hat{\mu}(\Sigma^{\leq n})}{2r} = \frac{1}{2r} < \frac{1}{r}.$$

□

We note that we can replace $\hat{\mu}_{\leq n}$ by $\hat{\mu}_n$ in statement (2) of Lemma 3.3.8, but not in statement (1) because $\lambda n.\hat{\mu}(\Sigma^n)$ may not be increasing in general.

Let μ be a distribution. In analogy with the terminology of "polynomial on μ-average," we say that a function g is $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ ($\{\mu_{\geq n}\}_{n\in\mathbb{N}}$, resp.) if there exists a polynomial p such that g is p on average with respect to $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ ($\{\mu_{\geq n}\}_{n\in\mathbb{N}}$, resp.). The above lemma immediately implies the following proposition.

Proposition 3.3.9 *For a function g from Σ^* to $\mathbb{R}^{+\infty}$, g is polynomial on μ-average if and only if g is polynomial on average with respect to $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$.*

Proof. The proposition follows from Lemma 3.3.8 together with the fact that if t is a polynomial, then $\lambda z.t(dz)$ and $\lambda z.t(2z^3)$ are both polynomials. □

Under some reasonable constraints, we can further replace an input ensemble $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ in the above proposition by $\{\mu_{\geq n}\}_{n\in\mathbb{N}}$. Cai and Selman [19] first proposed an idea of restricting Levin's notion of *polynomial on μ-average* to obtain a better time-hierarchy theorem. The following proposition takes a different formulation but shows an essential part of a theorem by Cai and Selman.

Proposition 3.3.10 *Let g be a function from Σ^* to $\mathbb{R}^{+\infty}$ and let μ be a distribution. If $\lambda n.\hat{\mu}(\Sigma^{\geq n}) \in \Omega(n^{-k})$ for some integer $k > 0$, then the following statements are equivalent:*

1. *g is polynomial on μ-average.*
2. *g is polynomial on average with respect to $\{\mu_{\geq n}\}_{n\in\mathbb{N}}$.*

Proof. Let g be a function from Σ^* to $\mathbb{R}^+$. Assume that $\lambda n.\hat{\mu}(\Sigma^{\geq n}) \in \Omega(n^{-k})$ for some $k > 0$.

We first show that (2) implies (1). Assume that there exists a polynomial p such that, for every $n \in \mathbb{N}$ and every $r > 0$, $\hat{\mu}_{\geq n}(\{x \in \Sigma^{\geq n} \mid g(x) > p(|x| \cdot r)\}) < 1/r$. Set $n = 0$, and then we obtain

$$\hat{\mu}(\{x \mid g(x) > p(|x| \cdot r)\}) < \frac{1}{r}.$$

Thus, g is polynomial on μ-average.

Next we shall show the other implication. Assume that g is polynomial on μ-average; namely, for some appropriate polynomial p, $\hat{\mu}(\{x \mid g(x) > p(r \cdot |x|)\}) < 1/r$ holds for all real numbers $r > 0$. Without loss of generality, we may assume that p is increasing.

By the assumption for μ, we can assume that, for some constant $c > 0$, $\hat{\mu}(\Sigma^{\geq n}) \geq \frac{1}{cn^k}$ holds for all positive integers n. For simplicity, assume that c is an integer.

Let us define q as $q(z) = p(2c \cdot z^{k+3})$ for all z. Clearly q is a polynomial. In particular, by the monotonicity of p,

$$q(r \cdot m) \geq p(m \cdot 2crm^{k+2})$$

for all numbers $r, m \geq 0$.

Now let S be the set of all natural numbers n for which μ_n is defined:

$$\hat{\mu}_{\geq n}(\{x \in \Sigma^{\geq n} \mid g(x) > q(r \cdot |x|)\})$$

$$
\begin{aligned}
&= \frac{1}{\hat{\mu}(\Sigma^{\geq n})} \cdot \hat{\mu}(\{x \in \Sigma^{\geq n} \mid g(x) > p(|x| \cdot 2cr|x|^{k+2})\}) \\
&= \sum_{m: m \in S \wedge m \geq n} \frac{1}{\hat{\mu}(\Sigma^{\geq n})} \cdot \hat{\mu}(\{x \in \Sigma^{m} \mid g(x) > p(|x| \cdot 2crm^{k+2})\}) \\
&\leq \sum_{m: m \in S \wedge m \geq n} \frac{1}{\hat{\mu}(\Sigma^{\geq m})} \cdot \frac{1}{2crm^{k+2}} \\
&\leq \sum_{m \geq n} cm^{k} \cdot \frac{1}{2crm^{k+2}} \\
&\leq \sum_{m=1}^{\infty} \frac{1}{2rm^{2}} = \frac{\pi^{2}}{12r} < \frac{1}{r}.
\end{aligned}
$$

Therefore, g is polynomial on average with respect to $\{\mu_{\geq n}\}_{n \in \mathbb{N}}$, and this completes the proof. □

3.3.3 Basic Properties

The notion of "t on μ-average," which we have introduced in the previous section, is essential in our average-case complexity theory. In this section, we shall discuss its fundamental properties which will be used freely in the later chapters.

The lemma below is the starting point.

Lemma 3.3.11 *Let k be a function on $\mathbb{N}$ such that $k(n) \geq 1$. For each $n > 0$, let $\{A_i^n \mid 1 \leq i \leq k(n)\}$ be a partition of Σ^n, i.e., $\Sigma^n = \bigcup_{i=1}^{k(n)} A_i^n$. Let g be a function from Σ^* to $\mathbb{R}^+$, t be a function on $\mathbb{R}^+$, and let μ be a distribution.*

1. *If t is increasing, then $\lambda x.t(|x|)$ is t on μ-average, where x runs over all strings.*
2. *If g is t on μ-average, then $g(x) \leq t(|x|/\hat{\mu}(x))$ for all x with $\hat{\mu}(x) > 0$.*
3. *Assume that t is increasing. If $g(\lambda) \leq t(0)$ and $g(x) \leq t(1/k(n)(n+1)\hat{\mu}(A_i^n))$ for all strings $x \in A_i^n$ if $\hat{\mu}(A_i^n) > 0$ for each $n > 0$ and each i with $1 \leq i \leq k(n)$, then g is t on μ-average.*

Proof. (1) If $0 < r < 1$, then $\hat{\mu}(\{x \mid t(|x|) > t(|x| \cdot r)\}) \leq 1 < \frac{1}{r}$. Assume $r \geq 1$. In this case, the set $\{x \mid t(|x|) > t(|x| \cdot r)\}$ is empty since t is increasing. Hence, $\hat{\mu}(\{x \mid t(|x|) > t(|x| \cdot r)\}) = 0 < \frac{1}{r}$. Therefore, $\lambda x.t(|x|)$ is t on μ-average.

(2) Assuming the contrary, suppose that there exists an x_0 such that $g(x_0) > t(|x_0|/\hat{\mu}(x_0))$ and $\hat{\mu}(x_0) > 0$. Let $r = 1/\hat{\mu}(x_0)$. Then, $\hat{\mu}(x_0) \leq \hat{\mu}(\{x \mid g(x) > t(|x| \cdot r)\}) < 1/r = \hat{\mu}(x_0)$, a contradiction.

(3) Assume that $g(\lambda) \leq t(0)$ and $g(x) \leq t(1/k(n)(n+1)\hat{\mu}(A_i^n))$ for all $x \in A_i^n$ if $\hat{\mu}(A_i^n) > 0$. Since t is increasing, $t(x) > t(y)$ implies $x > y$. We show that $\hat{\mu}(\{x \mid g(x) > t(|x| \cdot r)\}) < 1/r$. For each $r > 0$,

$$
\hat{\mu}(\{x \mid g(x) > t(|x| \cdot r)\}) \leq \sum_{n=1}^{\infty} \sum_{i=1}^{k(n)} \hat{\mu}(\{x \in A_i^n \mid t(1/k(n)(n+1)\hat{\mu}(A_i^n)) > t(n \cdot r)\})
$$

$$= \sum_{n=1}^{\infty}\sum_{i=1}^{k(n)} \hat{\mu}\left(\left\{ x \in A_i^n \;\middle|\; \frac{1}{k(n)(n+1)\hat{\mu}(A_i^n)} > n \cdot r \right\}\right)$$
$$= \sum_{n=1}^{\infty}\sum_{i=1}^{k(n)} \hat{\mu}\left(\left\{ x \in A_i^n \;\middle|\; \hat{\mu}(A_i^n) < \frac{1}{k(n)n(n+1)r} \right\}\right).$$

Notice that $\hat{\mu}(\{x \in A_i^n \mid \hat{\mu}(A_i^n) < \ell\}) < \ell$. Thus, we conclude that

$$\hat{\mu}(\{x \mid g(x) > t(|x| \cdot r)\}) < \sum_{n=1}^{\infty}\sum_{i=1}^{k(n)} \frac{1}{k(n)n(n+1)r} = \sum_{n=1}^{\infty} \frac{1}{n(n+1)r} = \frac{1}{r}$$

since $\sum_{n=1}^{\infty} \frac{1}{n(n+1)} = 1$. □

Lemma 3.3.11(2), for example, enables us to see that if g is polynomial on ν_{stand}-average, then g is exp-bounded. From the fact that $\hat{\nu}_{\mathrm{stand}}(x) \leq \frac{1}{2(|x|+1)^2} \cdot 2^{-|x|}$, it follows that

$$g(x) \leq c \cdot \left(\frac{|x|}{\hat{\nu}_{\mathrm{stand}}(x)}\right)^k + c \leq c \cdot \left(2|x|(|x|+1)^2 2^{|x|}\right)^k + c$$

for some constant $c > 0$.

Let us recall from Chapter 2 that *f majorizes g*, denoted by $f \geq g$, if and only if $f(x) \geq g(x)$ for all x.

Lemma 3.3.12 *Let f and g be any functions from Σ^* to $\mathbb{R}^{+\infty}$, t any function on $\mathbb{R}^+$, and μ any distribution. Assume that f majorizes g. If f is t on μ-average, then g is also t on μ-average.*

Proof. Assume that f is t on μ-average. Since $f(x) \geq g(x)$ for all x, it follows that $\{x \mid f(x) > c\} \subseteq \{x \mid g(x) > c\}$ for an arbitrary c. Then,

$$\hat{\mu}(\{x \mid g(x) > t(|x| \cdot r)\}) \leq \hat{\mu}(\{x \mid f(x) > t(|x| \cdot r)\}) < \frac{1}{r}.$$

Thus, g is t on μ-average. □

As we have seen in Subsection 3.3.1, Levin's notion of "polynomial on μ-average" is superior to the naive notion of "average polynomial" because the set of functions which are polynomial on μ-average is closed under algebraic operations, such as $+$, $\times$, max, and min. Here we show a more general claim that the set of functions which are $\mathcal{T}$ on μ-average is closed under such operations.

We say that $\mathcal{T}$ is *adequate* if, for any functions $t_1, t_2 \in \mathcal{T}$ and a constant $r > 0$, there exist functions $s_1, s_2, s_3 \in \mathcal{T}$ such that $s_1(x) \geq t_1(2x) + t_2(2x)$, $s_2(x) \geq (t_1(x))^r$, and $s_3(x) \geq r \cdot t_1(x)$ for all x.

Proposition 3.3.13 **[36]** *Let $\mathcal{T}$ be an adequate set of functions, let μ be a distribution, let f, g be functions from Σ^* to $\mathbb{R}^+$, and let r be a positive real number. If f, g are $\mathcal{T}$ on μ-average, then so are* $\max\{f, g\}$, $\min\{f, g\}$, f^r, $r \cdot f$, $f \times g$, *and* $f + g$, *where* $f^r(x) = (f(x))^r$.

Proof. Assume that $\hat{\mu}(\{x \mid f(x) > p(|x| \cdot r)\}) < 1/r$ and $\hat{\mu}(\{x \mid g(x) > q(|x| \cdot r)\}) < 1/r$. For the case

$f \times g$, take a function $s \in \mathcal{T}$ such that $s(z) \geq p(2z) + q(2z)$ for all z. Then, we have

$$\begin{aligned}
&\hat{\mu}(\{x \mid f(x)g(x) > s(|x| \cdot r)\}) \\
&\quad \leq \hat{\mu}(\{x \mid f(x) > p(|x| \cdot 2r)\}) + \hat{\mu}(\{x \mid f(x) \leq p(|x| \cdot 2r) \wedge f(x)g(x) > s(|x| \cdot r)\}) \\
&\quad < \frac{1}{2r} + \hat{\mu}(\{x \mid g(x) > q(|x| \cdot 2r)\}) \\
&\quad < \frac{1}{2r} + \frac{1}{2r} = \frac{1}{r}.
\end{aligned}$$

Similarly, for the case f^r, take a function s such that $s(z) \geq (p(z))^r$; for the case $r \cdot f$, take s such that $s(z) \geq r \cdot p(z)$; and for the case $f + g$, take a function s such that $s(z) \geq p(2z) + q(2z)$. The case $\max\{f, g\}$ is derived from the case $f + g$ since $f + g$ majorizes $\max\{f, g\}$. Similarly, $\min\{f, g\}$ becomes $\mathcal{T}$ on μ-average because $\max\{f, g\}$ majorizes $\min\{f, g\}$. □

As for "polynomial on μ-average," we shall show in the following lemma that this notion is invariant to any application of polynomials. This lemma is valuable in later chapters.

We call an n-ary function f *increasing* (or *monotone*) if $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$ whenever $x_i \leq y_i$ for all i, $1 \leq i \leq n$.

Lemma 3.3.14 **cf. [36]** *Let $k \geq 1$ and let $g_1, \ldots, g_k$ be functions from Σ^* to $\mathbb{R}^+$ and let p be a k-ary increasing polynomial. If all $g_1, \ldots, g_k$ are polynomial on μ-average, then $\lambda x.p(g_1(x), \ldots, g_k(x))$ is also polynomial on μ-average.*

Proof. Assume that, for each i with $1 \leq i \leq k$, the function g_i is q_i on μ-average for some polynomial q_i. By definition, it follows that $\hat{\mu}(\{x \mid g_i(x) > q_i(|x| \cdot r)\}) < \frac{1}{r}$ for all positive real numbers r. Now let $s(z) = p(q_1(kz), \ldots, q_k(kz))$ for all z. Since p is increasing, $p(x_1, \ldots, x_k) > p(y_1, \ldots, y_k)$ implies that there exists a number i such that $x_i > y_i$. So, we have

$$\begin{aligned}
\hat{\mu}(\{x \mid p(g_1(x), \ldots, g_k(x)) > s(|x| \cdot r)\}) &= \hat{\mu}(\{x \mid p(g(x)) > p(q_1(|x| \cdot kr), \ldots, q_k(|x| \cdot kr))\}) \\
&\leq \hat{\mu}(\{x \mid \exists i[1 \leq i \leq k \wedge g_i(x) > q_i(|x| \cdot kr)]\}) \\
&\leq \sum_{i=1}^{k} \hat{\mu}(\{x \mid g_i(x) > q_i(|x| \cdot kr)\}) \\
&< \sum_{i=1}^{k} \frac{1}{k \cdot r} = \frac{1}{r}.
\end{aligned}$$

Therefore, $\lambda x.p(g_1(x), \ldots, g_k(x))$ is polynomial on μ-average. □

The following lemma is of a rather simpler form than Lemma 3.3.11(3), but it is helpful to show elsewhere that a given function is polynomial on μ-average.

Lemma 3.3.15 *Let c, d, k be positive constants. Let $\{A_i^n \mid n > 0, 1 \leq i \leq n\}$ be a partition of Σ^+ such that $\Sigma^n = \bigcup_{i=1}^{n} A_i^n$ for each $n > 0$. Assume that, for almost all n, for all i with $1 \leq i \leq n$ and for all strings*

$x \in A_i^n$, $g(x) \leq c \cdot n^d + (c/n^2\hat{\mu}(A_i^n))^k$. Then, g is polynomial on μ-average.

Proof. Let us assume that g satisfies the above condition. For every x, let $g'(x) = \min\{g(x), c \cdot |x|^d\}$ and $g''(x) = g(x) - g'(x)$. Since $\lambda x.c|x|^d$ majorizes g', by Lemma 3.3.12, g' can be shown to be polynomial on μ-average.

We next show that g'' is also polynomial on μ-average. Let n_0 be the minimal integer such that, for all $n \geq n_0$, for all i with $1 \leq i \leq k(n)$ and for all x in A_i^n, $g''(x) \leq (c/n^2\hat{\mu}(A_i^n))^k$. Also let $b = \max\{g''(x) \mid |x| < n_0\}$. Note that

$$\left(\frac{c}{n^2\hat{\mu}(A_i^n)}\right)^k = \left(\frac{2c}{2n^2\hat{\mu}(A_i^n)}\right)^k \leq \left(\frac{2c}{n(n+1)\hat{\mu}(A_i^n)}\right)^k.$$

Now consider the polynomial p such that $p(z) = (2c \cdot z)^k + b$. It is obvious by our definition that $g''(\lambda) \leq p(0)$ and $g''(x) \leq p(1/n(n+1)\hat{\mu}(A_i^n))$ for all $x \in A_i^n$. By Lemma 3.3.11(3), g'' is polynomial on μ-average.

By Lemma 3.3.13, $g' + g''$ is also polynomial on μ-average. Since $g = g' + g''$, the proof is completed. □

We call a set $\mathcal{T}$ *suitable* if, for every $t \in \mathcal{T}$, every $c \in \mathbb{N}$, and every polynomial p, there are functions $s_1, s_2, s_3 \in \mathcal{T}$ such that $s_1(z) \geq t(cz)$, $s_2(z) \geq t(p(z))$, and $s_3(x) \geq t(z) + c$ for all x.

Lemma 3.3.16 *Let f be a p-bounded, p-honest function on Σ^*, and let g be a function from Σ^* to $\mathbb{R}^+$. Let $\mathcal{T}$ be a suitable set of increasing functions on $\mathbb{R}^+$. The function $g \circ f$ is $\mathcal{T}$ on μ-average if and only if g is $\mathcal{T}$ on $\mu_{f^{-1}}$-average.*

Proof. (If – part) Assume that g is t on $\mu_{f^{-1}}$-average for some $t \in \mathcal{T}$. Take an increasing polynomial p such that $|f(x)| \leq p(|x|)$ for all x, and an integer c_0 such that $g(\lambda) \leq c_0$. Moreover, we let function $t' \in \mathcal{T}$ be such that $t'(z) \geq t(3p(z)^3 z) + c_0$ for all z. Such a function exists because of the closure property of $\mathcal{T}$. Note that, for $x \in \Sigma^+$ and $r \geq 1$, $t(3p(|x| \cdot r)^3 \cdot |x|r) \geq t(3|f(x)|^3 r)$. We can show that $g \circ f$ is t' on μ-average. For the sake of convenience, let $A_\lambda = \{x \mid f(x) \neq \lambda\}$.

For each $r \geq 1$,

$$\begin{aligned}
\hat{\mu}(\{x \mid g \circ f(x) > t'(|x| \cdot r)\}) &\leq \hat{\mu}(\{x \in A_\lambda \mid \exists y[f(x) = y \wedge g(y) > t(3p(|x| \cdot r)^3 |x| r)]\}) \\
&\leq \hat{\mu}(\bigcup_{n=1}^{\infty} \bigcup_{y:|y|=n} \{x \in A_\lambda \mid f(x) = y \wedge g(y) > t(3|y|^3 r)\}) \\
&= \sum_{n=1}^{\infty} \sum_{y:|y|=n} \hat{\mu}(\{x \mid f(x) = y \wedge g(y) > t(|y| \cdot 3n^2 r)\}).
\end{aligned}$$

By the definition of $\mu_{f^{-1}}$, $\hat{\mu}(\{x \mid f(x) = y \wedge Q(y)\}) = \hat{\mu}_{f^{-1}}(\{y \mid Q(y)\})$ for any property Q. Thus, we get

$$\begin{aligned}
\hat{\mu}(\{x \mid g \circ f(x) > t'(|x| \cdot r)\}) &\leq \sum_{n=1}^{\infty} \hat{\mu}_{f^{-1}}(\{y \in \Sigma^n \mid g(y) > t(|y| \cdot 3n^2 r)\}) \\
&\leq \sum_{n=1}^{\infty} \frac{1}{3n^2 \cdot r} = \frac{\pi^2}{18r} < \frac{1}{r}.
\end{aligned}$$

(Only if – part) Assume that $g \circ f$ is t on μ-average for some $t \in \mathcal{T}$. Take an increasing polynomial p such that $|x| \leq p(|f(x)|)$ for all x because of the p-honesty of f. Let us take a function $t' \in \mathcal{T}$ such that $t'(z) \geq t(p(z)z) + c_0$ for all z, where c_0 is the same constant as defined above. We also take A_λ as above. Note that, for $x \in A_\lambda$ and $r \geq 1$,

$$t(p(|f(x)| \cdot r)|f(x)| \cdot r) \ \geq \ t(p(|f(x)|) \cdot r) \ \geq \ t(|x| \cdot r).$$

We show below that g is t' on $\mu_{f^{-1}}$-average.

$$\begin{aligned} \hat{\mu}_{f^{-1}}(\{y \mid g(y) > t'(|x| \cdot r)\}) &\leq \hat{\mu}(\{x \in A_\lambda \mid \exists y[f(x) = y \wedge g(y) > t(p(|y| \cdot r)|y| \cdot r)]\}) \\ &= \hat{\mu}(\{x \in A_\lambda \mid g(f(x)) > t(p(|f(x)| \cdot r)|f(x)| \cdot r)\}) \\ &\leq \hat{\mu}(\{x \mid g \circ f(x) > t(|x| \cdot r)\}) \ < \ \frac{1}{r}. \end{aligned}$$

□

3.3.4 Different Characterization

As stated before, our definition of "polynomial on μ-average" is motivated by Schapire's definition, and at first glance, it appears to be different from Levin's definition. In the first part of this section, we shall prove by a slightly more general argument that both definitions are equivalent. For this purpose, we introduce a "good" set of functions below.

For a set $\mathcal{T}$ of functions on $\mathbb{R}^+$, we call $\mathcal{T}$ *good* if, for every $s \in \mathcal{T}$ and every constant $c \in \mathbb{N}$, there exist functions $s', s'' \in \mathcal{T}$ such that $s(cx) \leq s'(x)$ and $s(x^2) \leq s''(x)$ for all x. For example, the set of increasing polynomials is a good set of convex functions.

Proposition 3.3.17 *Let g be a function from Σ^* to $\mathbb{R}^+$ and let $\mathcal{T}$ be a good set of strictly increasing functions on $\mathbb{R}^+$. The function g is $\mathcal{T}$ on μ-average if and only if the expectation $\sum_{x:|x|>0} |x|^{-1} \cdot t^{-1}(g(x)) \cdot \hat{\mu}(x)$ converges for some function $t \in \mathcal{T}$.*

Proof. (Only if – part) Without loss of generality, assume that $g(\lambda) = 0$ and that g is t on μ-average for some $t \in \mathcal{T}$. Then it follows that, for any real number $r > 0$, $\hat{\mu}(\{x \mid g(x) > t(r^2 \cdot |x|)\}) < 1/r^2$. By our assumption of $\mathcal{T}$, we can define a strictly increasing function $t' \in \mathcal{T}$ such that $t(z^2) \leq t'(z)$ for all z. In particular, $t(r^2 \cdot |x|) \leq t((r \cdot |x|)^2) \leq t'(r \cdot |x|)$. Hence, this implies that $\hat{\mu}(\{x \mid g(x) > t'(|x| \cdot r)\}) < 1/r^2$. Without loss of generality, we may let this t' be t in the rest of the proof.

This indicates that, for every integer $k > 0$, $\hat{\mu}(\{x \in \Sigma^+ \mid |x|^{-1}t^{-1}(g(x)) > k\}) < 1/k^2$. Using this inequality, we bound $\sum_{x:|x|>0} \frac{t^{-1}(g(x))}{|x|}\hat{\mu}(x)$ above by

$$\begin{aligned} \sum_{x:|x|>0} \frac{t^{-1}(g(x))}{|x|}\hat{\mu}(x) &\leq \sum_{k=1}^{\infty} k \cdot \hat{\mu}(\{x \in \Sigma^+ \mid k-1 < \frac{t^{-1}(g(x))}{|x|} \leq k\}) \\ &= \sum_{k=1}^{\infty}\sum_{i=k}^{\infty} \hat{\mu}(\{x \in \Sigma^+ \mid i-1 < \frac{t^{-1}(g(x))}{|x|} \leq i\}) \end{aligned}$$

$$= \sum_{k=1}^{\infty} \hat{\mu}\left(\bigcup_{i=k}^{\infty}\left\{x \in \Sigma^+ \mid i-1 < \frac{t^{-1}(g(x))}{|x|} \leq i\right\}\right).$$

The last term is further bounded above by

$$\hat{\mu}\left(\left\{x \in \Sigma^+ \;\middle|\; \frac{t^{-1}(g(x))}{|x|} > 0\right\}\right) + \sum_{k=1}^{\infty} \hat{\mu}(\{x \in \Sigma^+ \mid \frac{t^{-1}(g(x))}{|x|} > k\})$$
$$< \quad 1 + \sum_{k=1}^{\infty} \frac{1}{k^2} \quad = \quad 1 + \frac{\pi^2}{6}.$$

(If – part) Conversely, assume that $\sum_{x:|x|>0} t^{-1}(g(x))|x|^{-1}\hat{\mu}(x) \leq N$ for some number $N \geq 1$ and some strictly increasing function $t \in \mathcal{T}$. Markov's Inequality (Lemma A.1) enables us to get the inequality that $\hat{\mu}(\{x \mid |x|^{-1}t^{-1}(g(x)) > r \cdot N\}) < 1/r$ for any real number $r > 0$. This yields $\hat{\mu}(\{x \mid g(x) > t(rN|x|)\}) < 1/r$. Hence, g is $\lambda x.t(Nx)$ on μ-average. Since there is a function $t' \in \mathcal{T}$ such that $t'(z) \geq t(Nz)$ for all z, we get the desired result that g is $\mathcal{T}$ on μ-average. □

For example, the set of polynomials, $\mathcal{T} = \{\lambda z.(z^k + d) \mid k, d \geq 0\}$, and the set of logarithms, $\mathcal{T} = \{\lambda z.(k \log z + d) \mid k, d \geq 0\}$, are both good sets of strictly increasing functions on $\mathbb{R}^+$ (recall our convention: $\log z$ is defined to be 0 whenever $0 < z < 1$).

In 1995, Impagliazzo [43] pointed out that Levin's definition is equivalent to the statement that the expectation of a function over all strings of length $\geq n$, is bounded above by a polynomial in n.

In the following theorem, we see Schapire's [88] and Impagliazzo's characterization [43].

Theorem 3.3.18 **[88, 43]** *Let μ be a distribution and let g be a function from Σ^* to $\mathbb{R}^{+\infty}$. Let S be the set of all natural numbers n for which $\mu_{\leq n}$ is defined. The following statements are equivalent:*

1. *g is polynomial on μ-average.*
2. *$\sum_{x:|x|>0} \frac{g(x)^\delta}{|x|} \cdot \hat{\mu}(x) < \infty$ for some real number $\delta > 0$.*
3. *For all $n \in S$, $\sum_{x:|x|\leq n} g(x)^\delta \cdot \hat{\mu}_{\leq n}(x) < c \cdot n + d$, where δ is a positive real number and $c, d \in \mathbb{N}$.*

Proof. Note that if there exists a string x such that $g(x) = \infty$, then we can redefine $g(x) = 0$ without changing the three conditions above. Hence, we may assume that $g(x) < \infty$ for all x in the following proof.

The equivalence between (1) and (2) comes from Proposition 3.3.17. We next show that (2) and (3) are equivalent.

We first prove that (2) implies (3). Assume that $\sum_{x:|x|>0} \frac{g(x)^\delta}{|x|}\hat{\mu}(x) < \infty$ for some real number $\delta > 0$. Consider a large integer $c > 0$ such that $\sum_{x:|x|>0} \frac{g(x)^\delta}{|x|}\hat{\mu}(x) \leq c$. Let n_0 be the minimal number in S, and assume that an integer $d > 0$ satisfies that $\hat{\mu}_{\leq n_0}(\Sigma^{\leq n_0}) \geq \frac{1}{d}$. Then, for each $n \in S$,

$$\sum_{x:0<|x|\leq n} g(x)^\delta \hat{\mu}_{\leq n}(x) \quad \leq \quad \sum_{x:0<|x|\leq n} \frac{n}{|x|} \cdot g(x)^\delta \cdot \frac{\hat{\mu}(x)}{\hat{\mu}_{\leq n}(\Sigma^{\leq n})}$$
$$\leq \quad \frac{n}{\hat{\mu}_{\leq n}(\Sigma^{\leq n})} \sum_{x:|x|>0} \frac{g(x)^\delta}{|x|} \cdot \hat{\mu}(x)$$

$$\leq \quad c \cdot \frac{n}{\hat{\mu}_{\leq n_0}(\Sigma^{\leq n_0})} \quad \leq \quad cd \cdot n.$$

Conversely, assume (3); namely, $\sum_{x:|x|\leq n} g(x)^\delta \cdot \hat{\mu}_{\leq n}(x) < c \cdot n + d$ for all $n \in S$. Notice that

$$\sum_{x:|x|=n} g(x)^\delta \cdot \hat{\mu}(x) \leq c \cdot n + d$$

since

$$\sum_{x:|x|=n} g(x)^\delta \cdot \hat{\mu}(x) \leq \sum_{x:0<|x|\leq n} g(x)^\delta \cdot \hat{\mu}_{\leq n}(x) \leq c \cdot n + d.$$

To simplify the description, set $E_0 = \{x \in \Sigma^+ \mid g(x)^{\delta/3} \leq |x|\}$ and $E_1 = \{x \in \Sigma^+ \mid g(x)^{\delta/3} > |x|\}$. It is important to note that if $x \in E_1$, then $g(x)^{2\delta/3} > |x|^2$. Thus, we have

$$\begin{aligned}\sum_{x:|x|>0} \frac{g(x)^{\delta/3}}{|x|} \cdot \hat{\mu}(x) &= \sum_{x\in E_0} \frac{g(x)^{\delta/3}}{|x|} \cdot \hat{\mu}(x) + \sum_{x\in E_1} \frac{g(x)^{\delta/3}}{|x|} \cdot \hat{\mu}(x) \\ &\leq \sum_{x\in E_0} \hat{\mu}(x) + \sum_{x\in E_1} \frac{g(x)^\delta}{|x| \cdot g(x)^{2\delta/3}} \cdot \hat{\mu}(x).\end{aligned}$$

The first term $\sum_{x\in E_0} \hat{\mu}(x)$ is obviously at most 1. Let T be the last term, and we shall focus on it below:

$$\begin{aligned}T &\leq \sum_{x\in E_1} \frac{g(x)^\delta}{|x|^3} \cdot \hat{\mu}(x) \\ &\leq \sum_{n=1}^{\infty} \frac{1}{n^3} \sum_{x\in\Sigma^*} g(x)^\delta \cdot \hat{\mu}(x) \\ &\leq \sum_{n=1}^{\infty} \frac{c \cdot n + d}{n^3} = \sum_{n=1}^{\infty} \left(\frac{c}{n^2} + \frac{d}{n^3}\right) < \infty.\end{aligned}$$

□

3.3.5 Random Functions

Randomized algorithms are a simple tool for solving problems "fast" on the average. To cope with the running time/space of randomized algorithms, we need a notion of *polynomial on μ-average* for random functions because the running time/space forms a random function which depends on random seeds produced by the inner coin flipping process of the randomized algorithm.

Gurevich [36] and Blass and Gurevich [12] formulated a notion of *polynomial on μ-average* for random functions from a random-input domain Γ by requiring the convergence of the expectation of the value $|x|^{-1}g(x,s)^\delta$, over all pairs (x,s) in Γ, with respect to its probability $\hat{\mu}(x)$ and conditional probability $U_\Gamma(x) \cdot 2^{-|s|}$. Our formulation is a modification of their definition, and it will be shown to be equivalent to theirs at the end of this section.

Definition 3.3.19 (t on μ-Average) Let t be a function on $\mathbb{R}^+$ and let $\mathcal{T}$ be a set of functions on $\mathbb{R}^+$. Let μ be a distribution and let Γ be a random-input domain. Let g be a random function from Γ to $\mathbb{R}^{+\infty}$.

1. A random function g is t *on* μ*-average* if $\hat{\mu}_\Gamma(\{(x,s) \mid f(x,s) > t(|x| \cdot r)\}) < 1/r$ for all real numbers $r > 0$.

2. A random function f is $\mathcal{T}$ *on* μ*-average* if there exists a function $t \in \mathcal{T}$ such that f is t on μ-average.

The above definition implies that if a random function f with a random-input domain Γ is polynomial on μ-average, then the same function f, viewed as a "regular" function with two arguments, is polynomial on μ_Γ-average, i.e., $\hat{\mu}_\Gamma(\{(x,s) \mid f(x,s) > p(|\langle x,s\rangle| \cdot r)\}) < 1/r$ for some polynomial p. The converse, however, does not hold in general.

A simple observation shows that $h(\lambda,s) \leq t(0)$ for all random seeds $s \in \Gamma(\lambda)$ unless $\hat{\mu}(\lambda) = 0$. This is seen as follows. For any positive real number r,

$$\hat{\mu}(\lambda) \cdot \mathbf{Pr}_s[h(\lambda,s) > t(0) \mid s \in \Gamma(\lambda)] \leq \hat{\mu}_\Gamma(\{(\lambda,s) \mid h(\lambda,s) > t(0)\}) < \frac{1}{r}.$$

As r approaches ∞, the probability $\mathbf{Pr}_s[h(\lambda,s) > t(0) \mid s \in \Gamma(\lambda)]$ goes to 0. Therefore, this probability must be 0.

The lemma below provides us with a simple and sufficient method for proving a random function to be $\mathcal{T}$ on μ-average. For the lemma, we must recall the definition of a conditional expectation of a random function.

Fix a random-input domain Γ. For a random function f from Γ to $\mathbb{R}^+$, the conditional expectation $E_s[g(x,s) \mid s \in \Gamma(x)]$ for each x is defined by

$$E_s[g(x,s) \mid s \in \Gamma(x)] = \sum_{s \in \Gamma(x)} g(x,s) \cdot U_\Gamma(x) \cdot 2^{-|s|}.$$

Lemma 3.3.20 *Let $\mathcal{T}$ be a good set of strictly increasing convex functions and let μ be a distribution. Let Γ be a random-input domain and let g be a random function from Γ to $\mathbb{R}^+$. If $\lambda x.E_s[g(x,s) \mid s \in \Gamma(x)]$ is $\mathcal{T}$ on μ-average, then g is $\mathcal{T}$ on μ-average.*

Proof. Let $t \in \mathcal{T}$ and assume that $\lambda x.E_s[g(x,s) \mid s \in \Gamma(x)]$ is t on μ-average; namely, $\hat{\mu}(\{x \mid E_s[g(x,s) \mid s \in \Gamma(x)] > t(|x| \cdot r)\}) < 1/r$ for all $r > 0$. We shall show that $\hat{\mu}_\Gamma(\{(x,s) \mid g(x,s) > t'(|x| \cdot r)\}) < 1/r$ for some $t' \in \mathcal{T}$.

Fix $r \geq 1$. Since t is strictly increasing, $\hat{\mu}(\{x \mid t^{-1}(E_s[g(x,s) \mid s \in \Gamma(x)]) > |x| \cdot 4r^2\}) < 1/4r^2 < 1/r$. By Jensen's inequality, since t^{-1} is a strictly increasing concave function, it follows that

$$E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)] \leq t^{-1}(E_s[g(x,s) \mid s \in \Gamma(x)]).$$

Hence, $\hat{\mu}(\{x \mid E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)] > |x| \cdot 4r^2\}) < 1/r$. Notice that $E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)]$ does not depend on s. Then, clearly we have

$$\begin{aligned} &\hat{\mu}_\Gamma(\{(x,s') \mid E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)] > |x| \cdot 2r\}) \\ &\qquad = \quad \hat{\mu}(\{x \mid E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)] > |x| \cdot 2r\}) < \frac{1}{2r}. \end{aligned}$$

Now let $D_r = \{x \mid E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)] \leq |x| \cdot 2r\}$. We then obtain

$$\begin{aligned}\hat{\mu}_\Gamma(\{(x,s) \mid t^{-1}(g(x,s)) > |x| \cdot 4r^2\}) \\ \leq \quad & \hat{\mu}_\Gamma(\{(x,s') \mid E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)] > |x| \cdot 2r\}) \\ & + \hat{\mu}_\Gamma(\{(x,s) \mid x \in D_r \wedge t^{-1}(g(x,s)) > |x| \cdot 4r^2\}).\end{aligned}$$

Let us consider the last two terms in the above inequality. The first term is bounded above by $1/2r$. The second, however, is bounded above by

$$\hat{\mu}_\Gamma(\{(x,s) \mid t^{-1}(g(x,s)) > 2r \cdot E_s[t^{-1}(g(x,s)) \mid s \in \Gamma(x)]\})$$

because of the assumption $x \in D_r$. Markov's Inequality (Lemma A.1) ensures that this term is bounded above by $1/2r$. Thus, $\hat{\mu}_\Gamma(\{(x,s) \mid t^{-1}(g(x,s)) > |x| \cdot 4r^2\}) < 1/r$.

Using the fact that t is strictly increasing, we obtain the conclusion that $\hat{\mu}_\Gamma(\{(x,s) \mid g(x,s) > t(|x| \cdot 4r^2)\}) < 1/r$. Now consider another function $t' \in \mathcal{T}$ such that $t(4x^2) \leq t'(x)$ for all x, and as a result, we obtain the inequality $\hat{\mu}_\Gamma(\{(x,s) \mid g(x,s) > t'(|x| \cdot r)\}) < 1/r$. Therefore, g is $\mathcal{T}$ on μ-average. □

Note that the lemma shows only a sufficient condition, and the converse of the lemma may not hold in general. However, if we instead use Gill's measure (in Section 2.4), then we can obtain a necessary condition.

For a random function f from Γ to $\mathbb{R}^{+\infty}$, let f^* be defined as

$$f^*(x) = \begin{cases} \min\{n \mid \mathbf{Pr}_s[f(x,s) \leq n \mid s \in \Gamma(x)] > \frac{1}{2}\} & \text{if one exists,} \\ \infty & \text{otherwise.} \end{cases}$$

Lemma 3.3.21 *Let f be a random function with a random-input domain Γ. If f is t on μ-average, then f^* is $\lambda z.t(2z)$ on μ-average.*

Proof. For each x, if $f^*(x) > t(|x| \cdot 2r)$, then $\mathbf{Pr}_s[f(x,s) \leq t(|x| \cdot 2r) \mid s \in \Gamma(x)] \leq 1/2$, because $f^*(x)$ is the minimal value k satisfying that $\mathbf{Pr}_s[f(x,s) \leq k \mid s \in \Gamma(x)] > 1/2$. Hence, it follows that

$$2 \cdot \mathbf{Pr}_s[f(x,s) > t(|x| \cdot 2r) \mid s \in \Gamma(x)] > 1.$$

Using this fact, we can estimate the value $\hat{\mu}(\{x \mid f^*(x) > t(|x| \cdot 2r)\})$ as follows:

$$\begin{aligned}\hat{\mu}(\{x \mid f^*(x) > t(|x| \cdot 2r)\}) \quad & \leq \quad 2 \cdot \sum_x \hat{\mu}(x) \cdot \mathbf{Pr}_s[f(x,s) > t(|x| \cdot 2r) \mid s \in \Gamma(x)] \\ & = \quad 2 \cdot \hat{\mu}_\Gamma(\{(x,s) \mid f(x,s) > t(|x| \cdot 2r)\}) \\ & < \quad 2 \cdot \frac{1}{2r} \quad = \quad \frac{1}{r}.\end{aligned}$$

□

As a particular case, if a random function f is polynomial on μ-average, then its associated function f^* is also polynomial on μ-average.

When $f(x,s)$ is given as of the form $\mathrm{Time}_M(x;s)$ for some bounded-error probabilistic Turing machine M, we can also replace $\Gamma(x)$ by $\Omega(x)$ in the above definition of f^* and the lemma.

Lemma 3.3.22 *Let Γ be any random-input domain whose elements are of the form (x,y,s), and let μ and η be any two distributions. Let h be a function from Σ^* to $\mathbb{R}^+$ which is polynomial on μ-average, g a random function from Γ to $\mathbb{R}^+$ which is polynomial on $\mu\times\eta$-average, and f a function on Σ^*. If $h(f(x))\cdot\hat{\eta}(f(x))\geq 1$ for all x and $\lambda x.|f(x)|$ is polynomial on μ-average, then $\lambda xs.g(x,f(x),s)$ is polynomial on μ-average.*

Proof. For simplicity, write ν for $\mu\times\eta$. Let us assume that a random function g is polynomial on ν-average. This implies the existence of an increasing polynomial p_g such that $\hat{\nu}_\Gamma(\{(x,y,s)\mid g(x,y,s) > p_g(r\cdot(|x|+|y|))\})<1/r$ for any positive real number r. Similarly, from the assumption that $\lambda x.|f(x)|$ is polynomial on μ-average, there exists an increasing polynomial p_f such that $\lambda x.|f(x)|$ is p_f on μ-average. Let us also assume that h is q on μ-average for some increasing polynomial q.

We then define the new random-input domain Γ' as $\Gamma'=\{(x,s)\mid(x,f(x),s)\in\Gamma\}$. To reach the desired result, it suffices to show that, for some polynomial p, $\hat{\mu}_{\Gamma'}(\{(x,s)\mid g(x,f(x),s)>p(|x|\cdot r)\})<1/r$. This polynomial p is of the following form:

$$p_g((z+p_f(3z))\cdot 5z\cdot p_f(3z)^2\cdot q(p_f(3z)\cdot 3z)).$$

Let us fix r, $r>0$, and define $D_r=\{x\in\Sigma^+\mid h(f(x))\leq q(|f(x)|\cdot 3r)\wedge|f(x)|\leq p_f(|x|\cdot 3r)\}$. Notice that, by the monotonicity of p_f, p_g, and q, if $x\in D_r$, then

$$\begin{aligned} p(|x|\cdot r) &\geq p_g((|x|+p_f(|x|\cdot 3r))\cdot 5r\cdot p_f(|x|\cdot 3r)^2\cdot q(p_f(|x|\cdot 3r)\cdot 3r)) \\ &\geq p_g((|x|+|f(x)|)\cdot 5r\cdot|f(x)|^2\cdot q(|f(x)|\cdot 3r)). \end{aligned}$$

It is not difficult to see that

$$\begin{aligned} &\hat{\mu}_{\Gamma'}(\{(x,s)\mid x\notin D_r\}) \\ &\quad\leq \hat{\mu}(\{x\mid h(f(x))>q(|f(x))|\cdot 3r)\})+\hat{\mu}(\{x\mid|f(x)|>p_f(x\cdot 3r)\}) \\ &\quad< \frac{2}{3r}. \end{aligned}$$

In the rest of the proof, we shall show that $\hat{\mu}_{\Gamma'}(\{(x,s)\mid x\in D_r\wedge g(x,f(x),s)>p(|x|\cdot r)\})<1/3r$.

$$\begin{aligned} &\hat{\mu}_{\Gamma'}(\{(x,s)\mid x\in D_r\wedge g(x,f(x),s)>p(|x|\cdot r)\}) \\ &\quad= \hat{\mu}_{\Gamma'}(\bigcup_{n=1}^{\infty}\bigcup_{y:|y|=n}\{(x,s)\mid x\in D_r\wedge f(x)=y\wedge g(x,y,s)>p(|x|\cdot r)\}) \\ &\quad= \sum_{n=1}^{\infty}\sum_{y:|y|=n}\hat{\mu}_{\Gamma'}(\{(x,s)\mid x\in D_r\wedge f(x)=y \\ &\qquad\qquad\wedge g(x,y,s)>p_g((|x|+|y|)\cdot 5r|y|^2q(3r|y|))\}). \end{aligned}$$

Using the assumptions $q(3r|f(x)|)\geq h(f(x))$ and $h(f(x))\cdot\hat{\nu}_\Gamma(x,f(x),s)\geq\hat{\mu}_\Gamma(x,s)$, we obtain

$$\hat{\mu}_{\Gamma'}(\{(x,s)\mid x\in D_r\wedge g(x,f(x),s)>p(|x|\cdot r)\})$$

$$\begin{aligned}
&\leq \sum_{n=1}^{\infty} \sum_{y:|y|=n} q(|y|\cdot 3r)\cdot \hat{\nu}_\Gamma(\{(x,y,s) \mid x\in D_r \wedge f(x)=y \\
&\qquad\qquad \wedge\, g(x,y,s) > p_g((|x|+|y|)\cdot 5rn^2 q(3rn))\}) \\
&\leq \sum_{n=1}^{\infty} q(3rn)\cdot \hat{\nu}_\Gamma(\{(x,y,s) \mid x\in D_r \wedge g(x,y,s) > p_g((|x|+|y|)\cdot 5rn^2 q(3rn))\}) \\
&\leq \sum_{n=1}^{\infty} \frac{q(3rn)}{5n^2\cdot q(3rn)} \;=\; \frac{\pi^2}{30} \;<\; \frac{1}{3r}.
\end{aligned}$$

□

At the beginning of this subsection, we mentioned the equivalence between our definition of *polynomial on μ-average* and the definition given by Gurevich [36], and by Blass and Gurevich [12]. We now present this equivalence in the following proposition.

Proposition 3.3.23 *Let Γ be a random-input domain and let g be a random function from Γ to $\mathbb{R}^+$. The following statements are equivalent:*

1. *g is polynomial on μ-average.*
2. *$\sum_{(x,s):|x|>0} \frac{g(x,s)^\delta}{|x|}\cdot U_\Gamma(x)\cdot \hat{\mu}(x)\cdot 2^{-|s|} < \infty$ for some constant $\delta > 0$.*

Proof. Similar to Theorem 3.3.18. □

3.4 A Notion of Domination

Another crucial notion introduced by Levin [60] is the the notion of *polynomial domination* among distributions. The *domination relation* will be used as a part of the *domination condition* which is an important ingredient of the polynomial-time reducibility among distributional decision problems in Chapter 5. Intuitively speaking, it ensures that if an algorithm is "fast on the average" for some distribution μ, then this algorithm is also "fast on the average" for all distributions which are dominated by μ. This section will define the *polynomial domination relation* and the *polynomial equivalence relation* and explore some fundamental properties as preparation for Chapter 5.

3.4.1 Domination Relations and Equivalence Relations

First we shall give a general definition of domination relations.

Definition 3.4.1 (Domination Relations) Let μ and ν be semi-distributions.

1. Let t and $\mathcal{T}$ be a function and a set of functions from Σ^* to $\mathbb{R}^+$, respectively. The semi-distribution ν *t-dominates* μ if $t(x)\cdot\hat{\nu}(x) \geq \hat{\mu}(x)$ for all $x\in\Sigma^*$, and ν *$\mathcal{T}$-dominates* μ if there exists a function $t'\in\mathcal{T}$ such that ν t'-dominates μ.

2. Let t and $\mathcal{T}$ be a function and a set of functions on $\mathbb{R}^+$, respectively. The distribution ν *average* t-*dominates* μ if there exists a function t', from Σ^* to $\mathbb{R}^+$, such that t' is t on μ-average and ν t'-dominates μ, and ν *average* $\mathcal{T}$-*dominates* μ if there exists a function t in $\mathcal{T}$ such that ν average t-dominates μ.

This definition enables us to consider polynomial domination relations and average-polynomial domination relations.

Definition 3.4.2 (Polynomial Domination) Let μ and ν be any two semi-distributions.

1. The semi-distribution μ *polynomially dominates* (p-dominates, for short) ν if there exists a p-bounded function t such that μ t-dominates ν. For brevity, the notation $\nu \preceq^{\mathrm{p}} \mu$ expresses that μ p-dominates ν.

2. The semi-distribution μ *average-polynomially dominates* (avp-dominates, for short) ν if there exists a polynomial t such that μ average t-dominates ν. The notation $\mu \preceq^{\mathrm{avp}} \nu$ means that μ avp-dominates ν.

In [36], polynomial domination and average-polynomial domination are called *domination* and *weak domination*, respectively.

As previously mentioned, polynomial domination relations were explicitly introduced by Levin [60] on his theory of average-case complexity as a certain type of *reducibility* between two distributions which measures the complexity of these distributions. In this sense, two distributions which dominate each other can be considered to have almost the same degree of complexity. We call them *equivalent*. Equivalence relations capture the closeness of two distributions and also give rise to an appropriate "approximation" between them.

We begin by giving a general definition of equivalence relations.

Definition 3.4.3 (Equivalence Relations) Let μ and ν be any distributions. Let $\mathcal{T}$ be a set of functions on $\mathbb{R}^+$.

1. The distribution μ is $\mathcal{T}$-*equivalent* to ν if μ $\mathcal{T}$-dominates ν and ν $\mathcal{T}$-dominates μ.

2. The distribution μ is *average* $\mathcal{T}$-*equivalent* to ν if μ average$\mathcal{T}$-dominates ν and ν average$\mathcal{T}$-dominates μ.

Definition 3.4.4 (Polynomial Equivalence) A semi-distribution μ is *polynomially equivalent* (p-equivalent, for short) to another semi-distribution ν if μ is t-equivalent to ν for some p-bounded function t; and μ is *average-polynomially equivalent* (avp-equivalent, for short) to ν if μ is average t-equivalent to ν for some polynomial t. We use the notation $\mu \approx^{\mathrm{p}} \nu$ to mean that μ is p-equivalent to ν, and the notation $\mu \approx^{\mathrm{avp}} \nu$ to mean that μ is avp-equivalent to ν.

As an example, let us consider FLAT, the set of flat distributions. This set is invariant with respect to p-equivalence relations. This is seen as follows. Since μ is p-equivalent to ν, it holds that $\frac{\hat{\nu}(x)}{p(x)} \leq \hat{\mu}(x) \leq p(x)\hat{\nu}(x)$ for some p-bounded p. Then, we have

$$|\hat{\mu}(x) - \hat{\nu}(x)| \leq \max\left\{ |p(x)\hat{\nu}(x) - \hat{\nu}(x)|, \left|\frac{\hat{\nu}(x)}{p(x)} - \hat{\nu}(x)\right| \right\} \leq \frac{p(x)-1}{2^{|x|^{\epsilon}}}.$$

Hence, $|\hat{\mu}(x) - \hat{\nu}(x)| \leq 2^{-|x|^{\epsilon'}}$ for some $\epsilon' > 0$, and in consequence, μ is flat.

Domination and equivalence relations are transitive.

Lemma 3.4.5 *Let μ_1, μ_2 and μ_3 be distributions, and let $\mathcal{T}$ be a set of functions on $\mathbb{R}^+$.*

1. *If μ_3 $\mathcal{T}$-dominates μ_2 and μ_2 $\mathcal{T}$-dominates μ_1, then μ_3 $\mathcal{T}$-dominates μ_1.*
2. *If μ_3 is $\mathcal{T}$-equivalent to μ_2 and μ_2 is $\mathcal{T}$-equivalent to μ_1, then μ_3 is $\mathcal{T}$-equivalent to μ_1.*

Proof. (1) Assume that there are functions p_1 and p_2 in $\mathcal{T}$ such that $p_1(x) \cdot \hat{\mu}_2(x) \geq \hat{\mu}_1(x)$ and $p_2(x) \cdot \hat{\mu}_3(x) \geq \hat{\mu}_2(x)$ for all x. To obtain the desired result, we set $p(x) = p_1(x) \cdot p_2(x)$. Then, we have

$$p(x) \cdot \hat{\mu}_3(x) = p_1(x) \cdot (p_2(x) \cdot \hat{\mu}_3(x)) \geq p_1(x) \cdot \hat{\mu}_2(x) \geq \hat{\mu}_1(x).$$

(2) Immediate from (1). □

As an immediate consequence of Lemma 3.3.14, we have seen that if f majorizes g and f is polynomial on μ-average, then g is also polynomial on μ-average. Below we shall show that if ν avp-dominates μ and f is polynomial on ν-average, then f becomes polynomial on μ-average.

A class $\mathcal{T}$ of functions is said to be *closed under composition with polynomials* if, for any function t and any polynomial p with integer coefficients, $t \in \mathcal{T}$ implies $\lambda x.t(p(x)) \in \mathcal{T}$.

Lemma 3.4.6 *Let μ and ν be distributions, $\mathcal{T}$ a set of functions on $\mathbb{R}^+$ which is closed under composition with polynomials, and h a function from Σ^* to $\mathbb{R}^+$.*

1. *If ν avp-dominates μ and h is $\mathcal{T}$ on ν-average, then h is $\mathcal{T}$ on μ-average.*
2. *Provided that ν is avp-equivalent to μ, h is $\mathcal{T}$ on ν-average if and only if h is $\mathcal{T}$ on μ-average.*

Proof. (1) Assume that $\mu \preceq^{\mathrm{avp}} \nu$, and h is t on ν-average for some function $t \in \mathcal{T}$. Choose a function q which is polynomial on μ-average such that, for all $x \in \Sigma^*$, $\hat{\nu}(x) \cdot q(x) \geq \hat{\mu}(x)$. First notice that if h is degenerative under ν, then h is also degenerative under μ since $\hat{\nu}(x) = 0$ implies $\hat{\mu}(x) = 0$ for any string x. By our assumption, $\hat{\nu}(\{x \mid h(x) > t(|x| \cdot r)\}) < 1/r$ for all $r > 0$. Since the set $\mathcal{T}$ is closed under composition with polynomials, we assume without loss of generality that $h(\lambda) \leq t(0)$, and therefore $\hat{\mu}(\{x | h(\lambda) > t(0)\}) = 0$. Since q is polynomial on μ-average, there exists a polynomial p such that $\hat{\mu}(\{x \mid q(x) > p(|x| \cdot r)\}) < 1/r$ for all $r > 0$.

Let μ_n and ν_n denote the conditional probability of strings of length n of μ and ν, respectively. Note that if $q(x) \leq p(|x| \cdot r)$ for a string x of length n, then $\hat{\mu}(x) \leq p(n \cdot r) \cdot \hat{\nu}(x)$ for all $n \in \mathbb{N}$ and $r > 0$. Now define g as $g(x) = t(4x^3 \cdot p(2x))$. Since $\mathcal{T}$ is closed under composition with polynomials, g is in $\mathcal{T}$. We note that, for all $n \in \mathbb{N}$ and all $r \in \mathbb{R}^+$, $g(n \cdot r) \geq t(n \cdot r^3 \cdot 4n^2 \cdot p(n \cdot 2r))$. To complete the proof, we should show that h is g on μ-average. This is seen as follows: for every real number $r \geq 1$,

$$
\begin{aligned}
&\hat{\mu}(\{x \mid h(x) > g(|x| \cdot r)\}) \\
&\quad \leq \quad \hat{\mu}(\{x \mid q(x) > p(|x| \cdot 2r)\}) + \hat{\mu}(\{x \mid q(x) \leq p(|x| \cdot 2r) \wedge h(x) > g(|x| \cdot r)\}) \\
&\quad < \quad \frac{1}{2r} + \sum_{n=1}^{\infty} p(2nr) \cdot \hat{\nu}(\{x \in \Sigma^n \mid q(x) \leq p(2nr) \wedge h(x) > g(nr)\}) \\
&\quad \leq \quad \frac{1}{2r} + \sum_{n=1}^{\infty} p(2nr) \cdot \hat{\nu}(\{x \mid h(x) > t(|x| \cdot 4n^2 r^3 \cdot p(2nr))\}) \\
&\quad < \quad \frac{1}{2r} + \sum_{n=1}^{\infty} \frac{p(2nr)}{4n^2 r^3 \cdot p(2nr)} \quad = \quad \frac{1}{2r} + \frac{\pi^2}{24r^3} \quad < \quad \frac{1}{r}.
\end{aligned}
$$

(2) Let us recall that ν is avp-equivalent to μ if and only if ν avp-dominates μ and μ avp-dominates ν. The result thus follows immediately from (1). □

Corollary 3.4.7 *Let h be any function from Σ^* to $\mathbb{R}^{+\infty}$. For distributions μ and ν, if ν avp-dominates μ and h is polynomial on ν-average, then h is also polynomial on μ-average.*

Corollary 3.4.7 implies that the avp-equivalence relations preserve the notion of *polynomial on average*. This fact motivates us to introduce the two new notions of *inclusions* and *equality* among sets of distributions. These notions will be used in later chapters.

Definition 3.4.8 Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two sets of distributions.

1. $\mathcal{F}_1$ *polynomially includes* (p-includes, for short) $\mathcal{F}_2$, symbolically $\mathcal{F}_1 \subseteq^{\mathrm{p}} \mathcal{F}_2$, if every distribution in $\mathcal{F}_1$ is p-equivalent to some distribution in $\mathcal{F}_2$. Similarly, $\mathcal{F}_1$ *average polynomially includes* (avp-includes, for short) $\mathcal{F}_2$, symbolically $\mathcal{F}_1 \subseteq^{\mathrm{avp}} \mathcal{F}_2$, if every distribution in $\mathcal{F}_1$ is avp-equivalent to some distribution in $\mathcal{F}_2$.

2. $\mathcal{F}_1$ is *polynomially equal* (p-equal, for short) to $\mathcal{F}_2$, symbolically $\mathcal{F}_1 \cong^{\mathrm{p}} \mathcal{F}_2$, if $\mathcal{F}_1 \subseteq^{\mathrm{p}} \mathcal{F}_2$ and $\mathcal{F}_2 \subseteq^{\mathrm{p}} \mathcal{F}_1$. Similarly, $\mathcal{F}_1$ is *average polynomially equal* (avp-equal, for short) to $\mathcal{F}_2$, symbolically $\mathcal{F}_1 \cong^{\mathrm{avp}} \mathcal{F}_2$, if $\mathcal{F}_1 \subseteq^{\mathrm{avp}} \mathcal{F}_2$ and $\mathcal{F}_2 \subseteq^{\mathrm{avp}} \mathcal{F}_1$.

Lemma 3.4.9 *The relations $\subseteq^{\mathrm{p}}$, $\subseteq^{\mathrm{avp}}$, $\cong^{\mathrm{p}}$, and $\cong^{\mathrm{avp}}$ are reflexive and transitive. The relations $\cong^{\mathrm{p}}$ and $\cong^{\mathrm{avp}}$ are symmetrical.*

Proof. By Lemma 3.4.5 and the definitions. □

3.4.2 Fundamental Properties

The domination relations are of importance in average polynomial-time computation. This subsection will explore properties of domination relations in relation to functional composition, and prepare the ground for *domination conditions* which will be introduced and cultivated in Chapter 5.

Let us begin with several important properties. Recall that, for a function f and a distribution μ, the notation $\mu_{f^{-1}}$ denotes the default distribution defined by its probability $\hat{\mu}_{f^{-1}}(x) = \hat{\mu}(f^{-1}(x))$.

Lemma 3.4.10 *Let μ and ν be two distributions and let f be a p-honest, p-bounded function on Σ^*.*

1. *If $\mu \preceq^{\mathrm{p}} \nu$, then $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu_{f^{-1}}$.*
2. *If f is one-one, then $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$ if and only if $\mu \preceq^{\mathrm{p}} \nu \circ f$. Moreover, $\mu_{f^{-1}} \approx^{\mathrm{p}} \nu$ if and only if $\mu \approx^{\mathrm{p}} \nu \circ f$.*

Proof. (1) Assume that $\mu \preceq^{\mathrm{p}} \nu$. There is a p-bounded function s from Σ^* to $\mathbb{R}^+$ such that $\hat{\nu}(x) \cdot s(x) \geq \hat{\mu}(x)$. Without loss of generality, we assume that $s(x) \geq 1$ for all x. Consider an increasing polynomial p such that $s(x) \leq p(|x|)$ for all x. Since f is p-honest, there is an increasing polynomial q such that $|x| \leq q(|f(x)|)$ for all x. Now we let $t(y) = \max\{s(x) \mid f(x) = y\}$. It is clear from the definition that, for every y, there is an element $x_y \in f^{-1}(y)$ such that $t(y) \leq s(x_y)$. For this x_y, we have

$$t(y) \leq s(x_y) \leq p(|x_y|) \leq p(q(|f(x_y)|)) = p(q(|y|)) = p \circ q(|y|).$$

Since $p \circ q$ is again a polynomial, t should be p-bounded. Hence,

$$\begin{aligned} \hat{\nu}_{f^{-1}}(y) \cdot t(y) &= \sum_{x:f(x)=y} \hat{\nu}(x) \cdot t(y) \geq \sum_{x:f(x)=y} \hat{\nu}(x) \cdot s(x) \\ &\geq \sum_{x:f(x)=y} \hat{\mu}(x) = \hat{\mu}_{f^{-1}}(y). \end{aligned}$$

(2) Let f be a one-one, p-honest, p-bounded function on Σ^*. Assume that $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$. Take a p-bounded function p from Σ^* to $\mathbb{R}^+$ such that $\hat{\mu}(f^{-1}(y)) \leq \hat{\nu}(y) \cdot p(|y|)$ for all y. Write $x = f^{-1}(y)$. Then, since $y = f(x)$, we have $\hat{\mu}(x) \leq \hat{\nu}(f(x)) \cdot p(f(x))$. Let $q(x) = p(f(x))$. Notice that q is p-bounded because f is p-bounded. Thus, we get $\mu \preceq^{\mathrm{p}} \nu \circ f$.

Conversely, assume that $\mu \preceq^{\mathrm{p}} \nu \circ f$. There is an increasing polynomial p such that $\hat{\mu}(x) \leq \hat{\nu}(f(x)) \cdot p(|x|)$ for all x. Since f is p-honest, $|x| \leq s(|f(x)|)$ for some polynomial s. Write $y = f(x)$. Then, $x = f^{-1}(y)$ because f is one-one. Let $q(z) = p \circ f^{-1}(z)$, and then q is p-bounded, because

$$q(y) = p(|f^{-1}(y)|) = p(|x|) \leq p(s(|f(x)|)) = p(s(|y|)).$$

Hence, $\hat{\mu}(f^{-1}(y)) \leq \hat{\nu}(y) \cdot q(y)$ for all y. This yields the desired result $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$.

The final claim can be proved in a similar fashion. □

Lemma 3.4.11 *Let f be a function on Σ^* and let μ and ν be distributions. The following statements are equivalent:*

1. *There exists a p-bounded (polynomial on μ-average, resp.) function from Σ^* to $\mathbb{R}^+$ such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*

2. *There exists a semi-distribution η such that $\mu \preceq^{\mathrm{p}} \eta$ ($\mu \preceq^{\mathrm{avp}} \eta$, resp.) and $\hat{\nu} \geq \hat{\eta}_{f^{-1}}$.*

Proof. We shall show the implication from (2) to (1). Assume that $\mu \preceq^{\mathrm{p}} \eta$ and $\hat{\nu}(y) \geq \hat{\eta}_{f^{-1}}(y)$ for all y. Choose a function p such that $p(x) \cdot \hat{\eta}(x) \geq \hat{\mu}(x)$ for all x. Without loss of generality, we assume that $p(x) > 1$. Since $\hat{\eta}(x) \geq \frac{\hat{\mu}(x)}{p(x)}$,

$$\hat{\nu}(y) \geq \hat{\eta}_{f^{-1}}(y) = \hat{\eta}(\{x \mid x \in f^{-1}(y)\}) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}.$$

Conversely, we shall show that (1) implies (2). Let us assume that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y, where p is a p-bounded function. We may assume $p(x) > 1$ for all x. Let $\hat{\eta}(x) = \frac{\hat{\mu}(x)}{p(x)}$. Obviously, η is a semi-distribution. We then have

$$\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)} = \sum_{x \in f^{-1}(y)} \hat{\eta}(y) = \hat{\eta}_{f^{-1}}(y).$$

□

Lemma 3.4.12 *Let f be a p-honest, p-bounded function on Σ^*.*

1. *There exists a distribution η such that $\mu \preceq^{\mathrm{p}} \eta$ and $\hat{\nu} \geq \hat{\eta}_{f^{-1}}$ if and only if there exists a p-bounded function p from Σ^* to $\mathbb{R}^+$ such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*

2. *Assume that f is one-one. There exists a distribution η such that $\mu \preceq^{\mathrm{avp}} \eta$ and $\hat{\nu} \geq \hat{\eta}_{f^{-1}}$ if and only if there exists a function p which is polynomial on μ-average such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*

Proof. (Only if – part) Assume that $\mu \preceq^{\mathrm{p}} \eta$ and $\hat{\mu}(y) \geq \hat{\eta}_{f^{-1}}(y)$ for all y. As we have seen, the last inequality can be replaced by the corresponding equality.

We note that the inequality $\hat{\nu} \geq \hat{\eta}_{f^{-1}}$ in (iii) can be replaced by the equation $\hat{\nu} = \hat{\eta}_{f^{-1}}$, since if $\hat{\nu}(y_0) > \hat{\eta}_{f^{-1}}(y_0)$ for some y_0, then $\sum_y \hat{\nu}(y) = \sum_{y: y \neq y_0} \hat{\nu}(y) + \hat{\nu}(y_0) > \sum_{y: y \neq y_0} \hat{\eta}_{f^{-1}}(y) + \hat{\eta}_{f^{-1}}(y_0) = 1$, a contradiction. Hence, we may assume that $\hat{\mu}(y) = \hat{\eta}_{f^{-1}}(y)$ for all y.

There is a p-bounded function p such that $p(x) \cdot \hat{\eta}(x) = \hat{\mu}(x)$ for all x. In particular, if $p(x) = 0$, then reset $p(x)$ to be 1 without changing the equation since $\hat{\mu}(x) = 0$. So, we can assume that $p(x) > 0$ for all x. Then,

$$\hat{\nu}(y) = \sum_{x \in f^{-1}(y)} \hat{\eta}(x) = \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}.$$

(If – part) Assume that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all x, where p is a p-bounded function from Σ^* to $\mathbb{R}^+$. By Lemma 3.4.13, this is equivalent to $p'(y) \cdot \hat{\nu}(y) \geq \hat{\mu}_{f^{-1}}(y)$ for some p'. Consider p'' such that $p''(y) \cdot \hat{\nu}(y) = \hat{\mu}_{f^{-1}}(y)$ for all y. As above, we can assume that $p''(x) > 0$ for all x. Let $p_f(x) = p''(f(x))$. Then, we have $\hat{\nu}(y) = \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p_f(x)}$. Now define η as $\hat{\eta}(x) = \frac{\hat{\mu}(x)}{p_f(x)}$ for each x. Clearly we have $p_f(y) \cdot \hat{\eta}(y) = \hat{\mu}(y)$ and $\hat{\nu}(y) = \sum_{x \in f^{-1}(y)} \hat{\eta}(x) = \hat{\eta}_{f^{-1}}(y)$. □

Lemma 3.4.13 **[36]** *Let f be a p-honest, p-bounded function on Σ^*, and let μ and ν be distributions.*

1. *$\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$ if and only if there exists a p-bounded function p from Σ^* to $\mathbb{R}^+$ such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*
2. *Assume that f is one-one. Then, $\mu_{f^{-1}} \preceq^{\mathrm{avp}} \nu$ if and only if there exists a function p which is polynomial on μ-average such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*

Proof. We shall prove both claims simultaneously because the main difference is the condition on p.

(Only if – part). Assume that $p(y) \cdot \hat{\nu}(y) \geq \hat{\mu}_{f^{-1}}(y)$ for all y. We can assume without loss of generality that $p(y) > 0$ for all y. Define $p'(x) = p(f(x))$ for each x. If p is p-bounded, then p' is also p-bounded since f is p-bounded and $|p'(x)| = |p(f(x))| \leq q(|f(x)|)$ for some polynomial q. If p is polynomial on $\mu_{f^{-1}}$-average, Lemma 3.3.16 infers that p' is polynomial on μ-average. For $y \in \mathrm{ran}(f)$,

$$\hat{\nu}(y) \geq \frac{1}{p(y)} \hat{\mu}(\{x \mid f(x) = y\}) = \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(y)} = \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(f(x))} = \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p'(x)}.$$

(If – part). Assume that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for some p-bounded function p. Let $p'(y) = \max\{p(x) \mid x \in f^{-1}(y)\}$. If p is p-bounded, then p' is also p-bounded, since f is p-honest. If p is polynomial on μ-average, then using the fact that f is one-one, Lemma 3.3.16 again shows that p' is polynomial on $\mu_{f^{-1}}$-average. Then,

$$p'(y) \cdot \hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{p'(y)\hat{\mu}(x)}{p(x)} \geq \sum_{x \in f^{-1}(y)} \hat{\mu}(x) = \hat{\mu}_{f^{-1}}(y).$$

□

The following lemma is a polynomial-equivalence version of Lemma 3.4.13.

Lemma 3.4.14 *Let f be a p-honest, p-bounded function on Σ^*, and let μ and ν be distributions.*

1. *$\mu_{f^{-1}} \approx^{\mathrm{p}} \nu$ if and only if there exist two p-bounded functions p and q from Σ^* to $\mathbb{R}^+$ such that $\sum_{x \in f^{-1}(y)} q(x)\hat{\mu}(x) \geq \hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*
2. *Assume that f is one-one. Then, $\mu_{f^{-1}} \approx^{\mathrm{avp}} \nu$ if and only if there exist two functions p and q which are polynomial on μ-average such that $\sum_{x \in f^{-1}(y)} q(x)\hat{\mu}(x) \geq \hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*

Proof. We shall show both claims at once. We assume that $q(x) \cdot \hat{\mu}_{f^{-1}}(x) \geq \hat{\nu}(x)$ and $p(x) \cdot \hat{\nu}(x) \geq \hat{\mu}_{f^{-1}}(x)$ for all x. Let $p'(x) = p(f(x))$ and let $q'(x) = q(f(x))$. By the proof of Lemma 3.4.13, we have $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p'(x)}$ for all y. The other direction follows similarly:

$$\hat{\nu}(y) \leq q(y) \cdot \hat{\mu}_{f^{-1}}(y) = \sum_{x \in f^{-1}(y)} q(y)\hat{\mu}(x) = \sum_{x \in f^{-1}(y)} q(f(x))\hat{\mu}(x) = \sum_{x \in f^{-1}(y)} q'(x)\hat{\mu}(x).$$

If q is p-bounded, then q' is also p-bounded since f is p-bounded. We show that q' is polynomial on μ-average if q is polynomial on ν-average. Assume that q is s_0 on ν-average for some increasing polynomial s_0. By our assumption, p is polynomial on $\mu_{f^{-1}}$-average. Take a polynomial s_1 witnessing the average-polynomiality of p. Since f is p-bounded, there exists an increasing polynomial t such that $|f(x)| \leq t(|x|)$ for all x. Let $D_r = \{y \mid p(y) \leq s_1(|y| \cdot 2r)\}$. Let $s(z) = s_0(t(z) \cdot s_1(2t(z) \cdot z) \cdot 3t(z)^2 \cdot z)$ for all z. Note that, for $x \in \Sigma^+$ and $r \geq 1$,

$$s(|f(x)| \cdot r) \geq s_0(t(|x| \cdot r) \cdot s_1(2t(|x| \cdot r) \cdot 4t(|x| \cdot r)^2 \cdot |x| \cdot r) \geq s_0(|y| \cdot s_1(2|y| \cdot r) \cdot 4|y|^2 \cdot r),$$

where $y = f(x)$. For $r \geq 1$,

$$\begin{aligned}
&\hat{\mu}(\{x \mid q'(x) > s(|x| \cdot r)\}) \\
&\quad = \hat{\mu}(\{x \mid \exists y[f(x) = y \wedge q(y) > s(|x| \cdot r)]\}) \\
&\quad \leq \hat{\mu}_{f^{-1}}(\{y \mid q(y) > s_0(|y| \cdot s_1(2|y| \cdot r) \cdot 4|y|^2 \cdot r)\}) \\
&\quad = \hat{\mu}_{f^{-1}}(\{y \mid p(y) > s_1(|y| \cdot 2r)\}) + \hat{\mu}_{f^{-1}}(\{y \in D_r \mid q(y) > s_0(|y| \cdot s_1(2|y| \cdot r) \cdot 4|y|^2 \cdot r)\}) \\
&\quad \leq \frac{1}{2r} + \sum_{n=1}^{\infty} \hat{\mu}_{f^{-1}}(\{y \in D_r \cap \Sigma^n \mid q(y) > s_0(|y| \cdot s_1(2nr) \cdot 4n^2 r)\}).
\end{aligned}$$

By domination $s_1(2nr) \cdot \hat{\nu}(y) \geq \hat{\mu}_{f^{-1}}(y)$, therefore the last term of the above inequalities is bounded by

$$\begin{aligned}
&\sum_{n=1}^{\infty} s_1(2nr) \cdot \hat{\nu}(\{y \mid q(y) > s_0(|y| \cdot s_1(2nr) \cdot 4n^2 r)\}) \\
&\qquad \leq \frac{1}{2r} + \sum_{n=1}^{\infty} \frac{s_1(2nr)}{s_1(2nr) \cdot 4n^2 r} = \frac{1}{2r} + \frac{\pi^2}{24r} < \frac{1}{r}.
\end{aligned}$$

□

3.4.3 Randomized Domination

We also introduce a randomized version of p-domination relations and avp-domination relations.

Definition 3.4.15 (Randomized domination) Let μ and ν be any two semi-distributions.

1. The semi-distribution ν is said to *randomly p-dominate* μ (rp-dominate, for short), symbolically $\mu \preceq^{\mathrm{rp}} \nu$, if there exist a random-input domain Γ and a random function p from Γ to $\mathbb{N}$ such that

   (i) the random function p is p-bounded; and

(ii) $p(x,s) \cdot \hat{\nu}(x,s) \geq \hat{\mu}_\Gamma(x,s)$ for all pairs $(x,s) \in \Gamma$.

2. The semi-distribution ν is said to *randomly average p-dominate* μ (avrp-dominate, for short), symbolically $\mu \preceq^{\mathrm{rp}} \nu$, if there exist a random-input domain Γ and a random function p from Γ to $\mathbb{N}$ such that

(i) the random function p is polynomial on μ-average; and

(ii) $p(x,s) \cdot \hat{\nu}(x,s) \geq \hat{\mu}_\Gamma(x,s)$ for all pairs $(x,s) \in \Gamma$.

To emphasize the random-input domain γ, we use the notations $\mu \preceq^{\mathrm{rp}}_\Gamma \nu$ and $\mu \preceq^{\mathrm{avrp}}_\Gamma \nu$.

By definition, it is immediate that if Γ is almost total, then $\mu \preceq^{\mathrm{p}} \nu$ ($\mu \preceq^{\mathrm{avp}} \nu$, resp.) implies $\mu \preceq^{\mathrm{rp}}_\Gamma \nu_\Gamma$ ($\mu \preceq^{\mathrm{avrp}}_\Gamma \nu_\Gamma$, resp.). We have shown as in Corollary 3.4.7 that if $\nu \preceq^{\mathrm{avp}} \mu$ and h is polynomial on ν-average, then h is also polynomial on μ-average. In the following lemma, we shall show an analogous result for avrp-domination relation.

Lemma 3.4.16 *Let μ and ν be two distributions. Let h be a random function from Γ to $\mathbb{R}^{+\infty}$, where Γ is a random-input domain. Assume that, for some polynomial q, $\hat{\nu}(\{(x,s) \mid h(x,s) > q(r \cdot |x|)\}) < 1/r$ holds for any real number $r > 0$. If $\mu \preceq^{\mathrm{avrp}}_\Gamma \nu$, then h is polynomial on μ-average.*

Proof. The proof is similar to that of Lemma 3.4.6(1). Assume that h is q on ν-average. Since $\mu \preceq^{\mathrm{avrp}}_\Gamma \nu$, there exists a random function p being polynomial on μ-average such that $p(x,s) \cdot \hat{\nu}(x,s) \geq \hat{\mu}_\Gamma(x,s)$ for all pairs $(x,s) \in \Gamma$. Let us consider an increasing polynomial q' such that p is q' on μ-average.

We now fix a positive real number $r \geq 1$. Let us define the set $D_r = \{(x,s) \in \Gamma \mid x \in \Sigma^+ \wedge p(x,s) \leq q'(|x| \cdot 2r)\}$. Let c_0 be an integer such that $c_0 \geq \max\{h(\lambda,s) \mid s \in \Gamma(\lambda)\}$. Such an integer exists (see the observation made after Definition 3.3.19). As the desired polynomial $\tilde{q}$, we set $\tilde{q}(z) = q(4z^2 \cdot q'(2z)) + c_0$ for all z. The probability $\hat{\mu}_\Gamma(\{(x,s) \mid h(x,s) > \tilde{q}(|x| \cdot r)\})$ is bounded by

$$\hat{\mu}_\Gamma(\{(x,s) \mid p(x,s) > q'(|x| \cdot 2r)\}) + \hat{\mu}_\Gamma(\{(x,s) \in D_r \mid h(x,s) > \tilde{q}(|x| \cdot r)\}).$$

The first term is obviously bounded above by $1/2r$. Let T_r be the second term. To compute T_r, we note that if $(x,s) \in D_r$, then $q(|x| \cdot r) \cdot \hat{\nu}(x,s) \geq \hat{\mu}_\Gamma(x,s)$. Thus, T_r is calculated as follows:

$$\begin{aligned}
T_r &\leq \sum_{n=1}^{\infty} \hat{\mu}_\Gamma(\{(x,s) \in D_r \mid x \in \Sigma^n \wedge h(x,s) > q(|x| \cdot 4rn^2 q'(2rn))\}) \\
&\leq \sum_{n=1}^{\infty} q'(2rn) \cdot \hat{\nu}(\{(x,s) \mid h(x,s) > q(|x| \cdot 4rn^2 q'(2rn))\}) \\
&\leq \sum_{n=1}^{\infty} \frac{q'(2rn)}{4rn^2 \cdot q'(2rn)} = \sum_{n=1}^{\infty} \frac{1}{4rn^2} = \frac{\pi^2}{24r} < \frac{1}{2r}.
\end{aligned}$$

□

3.5 Distributional Decision Problems

In contrast to worst-case complexity theory, average-case complexity theory deals with not only a problem D but also a distribution μ of instances. Such a pair (D, μ) is called by many researchers a *distributional problem* [109, 110], *randomized problem* [36], or *random problem* [60]. The distribution μ assigns to an instance the probability of its occurrence as an input to the problem D.

Definition 3.5.1 (Distributional Decision Problems) A *distributional (decision) problem* (D, μ) is a pair of a set D of strings and a distribution μ. A set of distributional problems is called an *average-case complexity class*.

This section will focus on "natural" average-case complexity classes and give their formal definitions. Of particular interest are two types of average-case complexity classes. Following their definitions, we shall discuss general separation and collapse results among these average-case complexity classes.

3.5.1 Average-Case Complexity Classes

Similar to worst-case complexity theory, we can consider "complexity classes" of distributional decision problems. One natural type of those classes is the combination of existing worst-case complexity classes $\mathcal{C}$ and sets $\mathcal{F}$ of distributions. Such a class was first introduced by Levin [60] as an average-case version of **NP**, and later Ben-David, Chor, Goldreich, and Luby [9] invented a general notation $\langle \mathcal{C}, \mathcal{F} \rangle$ to describe such classes. Here we slightly modify their notation and introduce an average-case complexity class $\mathrm{Dist}(\mathcal{C}, \mathcal{F})$.

Definition 3.5.2 (Average-Case Complexity Classes) cf. [9] Let $\mathcal{C}$ be a complexity class and $\mathcal{F}$ be a class of distributions. Let $\mathrm{Dist}(\mathcal{C}, \mathcal{F})$ be the set $\{(D, \mu) \mid D \in \mathcal{C}, \mu \in \mathcal{F}\}$.

To simplify the notation, we devise the following convention: whenever the set of *all* distributions is discussed, we use the symbol $*$ (asterisk) as the distribution parameter. For example, $\mathrm{Dist}(\mathbf{NP}, *)$ denotes the collection of all pairs (D, μ) such that D is an **NP** set and μ is *any* distribution.

Here we see two examples of distributional decision problems in $\mathrm{Dist}(\mathbf{NP}, *)$.

Example 3.5.3 A graph $G = (V, E)$ is called *3-colorable* if there exists a coloring c of G (i.e., a map $c : V \to \{0, 1, 2\}$) such that, for any two distinct vertices u and v, if $(u, v) \in E$, then $c(u) \neq c(v)$. The *randomized 3-colorability problem* $(\mathrm{3COL}, \mu_{\mathrm{3COL}})$ is defined as follows: let

$$\mathrm{3COL} = \{\langle G \rangle \mid \ G \text{ is a graph which is 3-colorable } \};$$

and let

$$\hat{\mu}_{\mathrm{3COL}}(\langle G \rangle) = \hat{\nu}_{\mathrm{tally}}(1^{\|V\|}) \cdot 2^{-\binom{n}{2}},$$

where $\langle G \rangle$ is an appropriate encoding of G. The distribution μ_{3COL} is best described by the following experiment: randomly choose the number of vertices and then randomly choose edges between pairs of

distinct vertices. Since 3COL belongs to **NP**, the problem $(3\mathrm{COL}, \mu_{3\mathrm{COL}})$ belongs to $\mathrm{Dist}(\mathbf{NP}, *)$.

Example 3.5.4 The *randomized 3 satisfiability problem* $(3\mathrm{SAT}, \mu_{\mathrm{SAT}})$ is defined as follows:

$$3\mathrm{SAT} = \{\langle \hat{p}_1, \hat{q}_1, \hat{r}_1 \rangle, \ldots, \langle \hat{p}_n, \hat{q}_n, \hat{r}_n \rangle \mid \text{ formula } \bigwedge_{i=1}^{n} (p_i \vee q_i \vee r_i) \text{ is satisfiable } \},$$

where all $\hat{p}_i$, $\hat{q}_i$, and $\hat{r}_i$ are strings which are reasonable codes of Boolean variables p_i, q_i, and r_i, respectively; and let μ_{SAT} be defined by its probability

$$\hat{\mu}_{\mathrm{SAT}}(\langle \hat{p}_1, \hat{q}_1, \hat{r}_1 \rangle, \ldots, \langle \hat{p}_n, \hat{q}_n, \hat{r}_n \rangle) = \hat{\nu}_{\mathrm{tally}}(1^n) \cdot \sum_{i=1}^{n} 2^{-(|\hat{p}_i|+|\hat{q}_i|+|\hat{r}_i|)}.$$

It is clear that $(3\mathrm{SAT}, \mu_{\mathrm{SAT}})$ belongs to $\mathrm{Dist}(\mathbf{NP}, *)$.

Another type of average-case complexity classes is more involved with *algorithmic computability in feasible time on average*. First of all, we shall give a general definition of *time- and space-bounded on average* for Turing machines. Notice that deterministic and nondeterministic Turing machines are special cases of alternating Turing machines. We define the time and space complexity of alternating Turing machines in an average-case setting.

Definition 3.5.5 (Time/Space Bounded on Average) Let M be an alternating oracle Turing machine and μ a distribution. Let A and S be any sets. Also let t and $\mathcal{T}$ be a function and a set of functions on $\mathbb{R}^+$, respectively.

1. The machine M with oracle A is *t-time bounded on μ-average* if the function $\lambda x.\mathrm{Time}_M^A(x)$ is t on μ-average, and it is *$\mathcal{T}$-time bounded on μ-average* if it is t-time bounded on μ-average for some $t \in \mathcal{T}$. The machine M is said to *recognize S in t-time ($\mathcal{T}$-time,* resp.*) on μ-average* if M is t-time (or $\mathcal{T}$-time, resp.) bounded on μ-average, and $S = L(M, A)$.

2. The machine M with oracle A is *t-space bounded on μ-average* if the function $\lambda x.\mathrm{Space}_M^A(x)$ is t on μ-average, and it is *$\mathcal{T}$-space bounded on μ-average* if it is t on μ-average for some $t \in \mathcal{T}$. The machine M is said to *recognize S in t-space ($\mathcal{T}$-space,* resp.*) on μ-average* if M is t-space (or $\mathcal{T}$-space, resp.) bounded on μ-average, and $S = L(M, A)$.

We note that, in the case that $\hat{\mu}(x) = 0$, the machine M does not *necessarily* halt on this particular input x because, according to our interpretation, the instance x does not *occur*. Nevertheless, this is not crucial in our theory. Schuler and Yamakami [97], for example, have considered only machines which always halt.

For a randomized Turing machine M, we must demand that the random function $\lambda x s.\mathrm{Time}_M(x; s)$ be t on μ-average.

Definition 3.5.6 (Time Bounded on Average) Let M be a randomized oracle Turing machine and μ a distribution. Let A and S be any two sets. Also let t and $\mathcal{T}$ be a function and a set of functions on $\mathbb{R}^+$,

respectively. The machine M with oracle A is called *t-time bounded on μ-average* if the random function $\lambda xs.\mathrm{Time}_M^A(x;s)$ is t on μ-average, and it is *$\mathcal{T}$-time bounded on μ-average* if it is t-time bounded on μ-average for some $t \in \mathcal{T}$. The machine M is said to *recognize S in t-time ($\mathcal{T}$-time,* resp.*) on μ-average* if M is t-time (or $\mathcal{T}$-time, resp.) bounded on μ-average, and $S = L(M, A)$.

We are especially interested in machines which are *polynomial-time/space bounded on μ-average*.

Definition 3.5.7 (Polynomial-Time/Space Bounded on Average) Let M be an oracle Turing machine with output tapes. Let A be a set and let f be a function. We say that M with oracle A is *polynomial-time (polynomial-space,* resp.*) bounded on μ-average* if M^A is $\mathcal{T}$-time ($\mathcal{T}$-space, respectively) bounded on μ-average for $\mathcal{T}$ being the set of all polynomials (i.e., any functions of the form of $\sum_{i=1}^{k} a_i \cdot x^i$, $a_i \in \mathbb{Z}$). The machine M is said to *compute f in polynomial-time (polynomial-space,* resp.*) on μ-average* if M is t-time (or $\mathcal{T}$-time, resp.) bounded on μ-average, and $f(x) = M^A(x)$ for all x.

Now we are ready to introduce the second type of average-case complexity complexity classes $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$.

Definition 3.5.8 (Average-Case Complexity Classes) cf. [97] Let t, s be functions on $\mathbb{N}$ and let $\mathcal{T}$ and $\mathcal{S}$ be sets of functions on $\mathbb{N}$. Also let $\mathcal{F}$ be a class of distributions. Time- and space-bounded *average-case complexity classes* are defined as follows:

1. $\mathrm{Aver}(\mathrm{DTIME}(t), \mathcal{F})$ is the collection of distributional decision problems (D, μ) such that $\mu \in \mathcal{F}$ and D is computable by some deterministic Turing machine in t-time on μ-average. Let $\mathrm{Aver}(\mathrm{DTIME}(\mathcal{T}), \mathcal{F}) = \bigcup_{t \in \mathcal{T}} \mathrm{Aver}(\mathrm{DTIME}(t), \mathcal{F})$.

2. $\mathrm{Aver}(\mathrm{NTIME}(t), \mathcal{F})$ is the collection of distributional decision problems (D, μ) such that $\mu \in \mathcal{F}$ and D is recognized by some nondeterministic Turing machine in t-time on μ-average. Let $\mathrm{Aver}(\mathrm{NTIME}(\mathcal{T}), \mathcal{F}) = \bigcup_{t \in \mathcal{T}} \mathrm{Aver}(\mathrm{NTIME}(t), \mathcal{F})$.

3. $\mathrm{Aver}(\mathrm{BPTIME}(t), \mathcal{F})$ is the collection of (D, μ) such that $\mu \in \mathcal{F}$ and D is recognizable by some bounded-error probabilistic Turing machine in t-time on μ-average. Let $\mathrm{Aver}(\mathrm{BPTIME}(\mathcal{T}), \mathcal{F}) = \bigcup_{t \in \mathcal{T}} \mathrm{Aver}(\mathrm{BPTIME}(t), \mathcal{F})$. Similarly, $\mathrm{Aver}(\mathrm{RTIME}(t), \mathcal{F})$ is the collection of (D, μ) such that $\mu \in \mathcal{F}$ and D is recognizable by some one-sided error, probabilistic Turing machine in t-time on μ-average. Let $\mathrm{Aver}(\mathrm{RTIME}(\mathcal{T}), \mathcal{F}) = \bigcup_{t \in \mathcal{T}} \mathrm{Aver}(\mathrm{RTIME}(t), \mathcal{F})$.

4. $\mathrm{Aver}(\mathrm{ATIME}(t), \mathcal{F})$ is the collection of (D, μ) such that $\mu \in \mathcal{F}$ and $D = L(M)$ for an alternating Turing machine M which is t-time bounded on μ-average. The class $\mathrm{Aver}(\mathrm{ATIME}^{\Sigma}(t, s), \mathcal{F})$ is the collection of (D, μ) such that $\mu \in \mathcal{F}$ and $D = L(M)$ for an alternating Turing machine M in s-time on μ-average which is t-alternation bounded, starting with an existential state. Similarly, $\mathrm{Aver}(\mathrm{ATIME}^{\Delta}(t, s), \mathcal{F})$ is defined by semi-deterministic alternating Turing machines. Let $\mathrm{Aver}(\mathrm{ATIME}(\mathcal{T}), \mathcal{F}) = \bigcup_{t \in \mathcal{T}} \mathrm{Aver}(\mathrm{ATIME}(t), \mathcal{F})$, and similarly $\mathrm{Aver}(\mathrm{ATIME}^{\Delta}(\mathcal{T}, \mathcal{S}), \mathcal{F})$ and $\mathrm{Aver}(\mathrm{ATIME}^{\Sigma}(\mathcal{T}, \mathcal{S}), \mathcal{F})$ are defined.

5. $\mathrm{Aver}(\mathrm{DSPACE}(s), \mathcal{F})$ is the collection of (D, μ) such that $\mu \in \mathcal{F}$ and D is computable by some deterministic Turing machine M in s-space on μ-average. Let $\mathrm{Aver}(\mathrm{DSPACE}(\mathcal{S}), \mathcal{F}) = \bigcup_{s \in \mathcal{S}} \mathrm{Aver}(\mathrm{DSPACE}(s), \mathcal{F})$.

Ben-David, Chor, Goldreich, and Luby [9] use the notation $\mathrm{AverDTime}(t(n))$ instead to denote our $\mathrm{Aver}(\mathrm{DTIME}(t), *)$ (also denoted by $\mathrm{AvDTime}(t(n))$ in [84]).

In what follows, we use abbreviations commonly used in worst-case complexity theory, such as **P** (= $\mathrm{DTIME}(n^{O(1)})$), **PSPACE** (= $\mathrm{DSPACE}(n^{O(1)})$), **RP** (= $\mathrm{RTIME}(n^{O(1)})$), etc. (see Section 2.5).

Recall the randomized 3-colorability problem $(\mathrm{3COL}, \mu_{\mathrm{3COL}})$. It is well known that 3COL is **NP**-complete (see e.g., [26]). However, Wilf [117] showed that the randomized 3-colorability problem can be solved by some deterministic algorithm in time polynomial on μ_{3COL}-average. Thus, $(\mathrm{3COL}, \mu_{\mathrm{3COL}})$ belongs to $\mathrm{Aver}(\mathbf{P}, *)$.

We have introduced five categories of fundamental average-case complexity classes $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$. We can extend our definition to classes which do not fall into those categories. One such extension is the *complement* of an average complexity class $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$.

Definition 3.5.9 (Complement Classes) For a complexity class $\mathcal{C}$ and a set $\mathcal{F}$ of distributions, the *complement* of $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ is denoted by $\mathrm{Aver}(\text{co-}\mathcal{C}, \mathcal{F})$ and is defined by the collection of all distributional decision problems (D, μ) such that $(\overline{D}, \mu)$ belongs to $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$, where $\overline{D} = \Sigma^* - D$.

Another extension is the intersection of two average-case complexity classes $\mathrm{Aver}(\mathcal{C}_1, \mathcal{F})$ and $\mathrm{Aver}(\mathcal{C}_2, \mathcal{F})$.

Definition 3.5.10 (Intersection Classes) For two complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$, and a set $\mathcal{F}$ of distributions, the *intersection* of $\mathrm{Aver}(\mathcal{C}_1, \mathcal{F})$ and $\mathrm{Aver}(\mathcal{C}_2, \mathcal{F})$ is denoted by $\mathrm{Aver}(\mathcal{C}_1 \cap \mathcal{C}_2, \mathcal{F})$ and is defined by the collection of all distributional problems (D, μ) such that $(D, \mu) \in \mathrm{Aver}(\mathcal{C}_1, \mathcal{F})$ and $(D, \mu) \in \mathrm{Aver}(\mathcal{C}_2, \mathcal{F})$.

For example, we can define the following average-case complexity classes: $\mathrm{Aver}(\text{co-}\mathbf{RP}, \mathcal{F})$, $\mathrm{Aver}(\text{co-}\mathbf{NP}, \mathcal{F})$, $\mathrm{Aver}(\mathbf{ZPP}, \mathcal{F})$, and $\mathrm{Aver}(\mathbf{NP} \cap \text{co-}\mathbf{NP}, \mathcal{F})$.

Remark 3.5.11 The notation "$\mathrm{Aver}(\cdot, \cdot)$" in which we use parentheses referring to *functions* might be misleading. The reader should resist the temptation to replace a class $\mathcal{C}$ in the notation "$\mathrm{Aver}(\mathcal{C}, \mathcal{F})$" with another *equivalent* class $\mathcal{C}'$ because the equality $\mathcal{C} = \mathcal{C}'$ in worst-case complexity theory does not always imply the equality $\mathrm{Aver}(\mathcal{C}, \mathcal{F}) = \mathrm{Aver}(\mathcal{C}', \mathcal{F})$ in average-case complexity theory.

We shall introduce a counterpart of the function class **FP** in worst-case complexity theory: the *average polynomial-time computable functions*. Recall that a *transducer* is a Turing machine equipped with an extra output tape for the purpose of computing a function on Σ^*.

Definition 3.5.12 (Average Time Computable Functions) If a function f on Σ^* is computed by a deterministic transducer which is polynomial-time bounded on μ-average, we say that f is *computable*

in time polynomial on μ-average. Let $\mathrm{Aver}(\mathbf{FP}, \mathcal{F})$ denote the collection of all pairs (f, μ), where μ is a distribution in $\mathcal{F}$, and f is a function computable in time polynomial on μ-average.

We shall take a quick glance at some fundamental relationships among average complexity classes. By the definition of *t on μ-average*, the class $\mathrm{Dist}(\mathrm{DTIME}(t), \mathcal{F})$ is clearly included in $\mathrm{Aver}(\mathrm{DTIME}(t), \mathcal{F})$. Similar inclusions obviously hold for other fundamental average complexity classes.

Proposition 3.5.13 *Let $\mathcal{C} \in \{\mathrm{DTIME}(t), \mathrm{NTIME}(t), \mathrm{BPTIME}(t), \mathrm{RTIME}(t), \mathrm{DSPACE}(t)\}$ for some increasing function t on $\mathbb{N}$, and let $\mathcal{F}$ be a set of distributions. Then, $\mathrm{Dist}(\mathcal{C}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathcal{C}, \mathcal{F})$.*

The next lemma follows immediately from Lemmas 3.4.6(1) and Corollary 3.4.7.

Lemma 3.5.14 *Let $\mathcal{F}$ be any set of distributions and assume that $\mu, \nu \in \mathcal{F}$. Let $\mathcal{C}$ be any of the following classes: **L**, **P**, co-**NP**, **NP**, **RP**, **BPP**, and **EXP**. If $(D, \nu) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$ and ν avp-dominates μ, then $(D, \mu) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$.*

Proof. Since the sets of polynomials, logarithms, and exponentials are all closed under composition with polynomials, the lemma immediately follows from Lemma 3.4.6. □

Similar simulation techniques show basic inclusion relationships among average complexity classes.

Theorem 3.5.15 *Let t be a function on $\mathbb{R}^+$.*

1. $\mathrm{Aver}(\mathrm{DTIME}(t), \mathcal{F}) \subseteq \mathrm{Aver}(\mathrm{RTIME}(t), \mathcal{F})$.
2. $\mathrm{Aver}(\mathrm{RTIME}(t), \mathcal{F}) \subseteq \mathrm{Aver}(\mathrm{NTIME}(t), \mathcal{F})$.
3. $\mathrm{Aver}(\mathrm{RTIME}(t), \mathcal{F}) \subseteq \mathrm{Aver}(\mathrm{BPTIME}(O(t)), \mathcal{F})$.
4. $\mathrm{Aver}(\mathrm{BPTIME}(t), \mathcal{F}) \subseteq \mathrm{Aver}(\mathrm{DSPACE}(O(\lambda x.t(2x))), \mathcal{F})$.
5. $\mathrm{Aver}(\mathrm{DSPACE}(t), \mathcal{F}) \subseteq \mathrm{Aver}(\mathrm{DTIME}(O(2^t)), \mathcal{F})$, *where 2^t means $\lambda x.2^{t(x)}$.*

Proof. We use a standard simulation technique to show the above inclusions of average-case complexity classes.

(1) Obviously, deterministic Turing machines are viewed as a special case of one-sided error probabilistic Turing machines.

(2) It suffices to note that one-sided error probabilistic Turing machines are nondeterministic Turing machines because of one-sided error probability. (This is not true for two-sided error probabilistic machines.)

(3) Suppose (D, μ) is in $\mathrm{Aver}(\mathrm{RTIME}(t), \mathcal{F})$. Take a one-sided error, probabilistic Turing machine M recognizing D in time t on μ-average. Let us consider the simulation machine N defined as follows:

begin randomized algorithm for N
  input x
  simulate M on input x
  if M accepts x **then accept**
  simulate M on input x
  if M accepts x **then accept else reject**
end.

Assume that $x \in D$. The error occurs if both simulations of M on x fail to reach any accepting configuration. Hence, the error probability of N on input x is at most $1/4$. Assume to the contrary that $x \notin D$. Because of one-sided error probability, there is no error occurring in the computation of N on x. Therefore, N has bounded-error probability.

By definition, the running time of N on x, $\mathrm{Time}_N(x)$, is bounded by a linear function in $\mathrm{Time}_M(x)$. Since M is t-time bounded on μ-average, the function $\lambda x.\mathrm{Time}_N(x)$ is $\lambda x.c(t(x)+1)$ on μ-average for some positive constant c.

(4) First we observe that any bounded-error probabilistic Turing machine M can be simulated deterministically using at most $O(\mathrm{Time}^*_M(x))$ tape squares. This is seen as follows. We simulate M on x by choosing its configurations one by one (i.e., using breadth-first search) and simulating the corresponding steps. Assume that we visit enough leaves of the computation tree so that we can determine whether M accepts or rejects x. We then quit the simulation and go into an accepting state if M accepts x, or else go into a rejecting state. This simulation needs tape space at most $c \cdot (\mathrm{Time}^*_M(x)+1)$ for some fixed constant $c > 0$.

Now suppose that (D,μ) is in $\mathrm{Aver}(\mathrm{BPTIME}(t),\mathcal{F})$. There exists a bounded-error probabilistic Turing machine M which recognizes D in time t on μ-average. Since $\lambda x.\mathrm{Time}_M(x)$ is t on μ-average, Lemma 3.3.22 implies that $\lambda x.\mathrm{Time}^*_M(x)$ is $\lambda x.t(2x)$ on μ-average. The claim follows from the observation above.

(5) Given a deterministic Turing machine M, which on input x runs using space $p(x)$, the number of possible configurations given by M on x is $O(2^{p(x)})$. Hence, the simulation of M on x takes $O(2^{p(x)})$ time. □

Generally speaking, the function $\lambda x.\mathrm{Time}_M(x)$ for a nondeterministic Turing machine M is not time-constructible; therefore, even if t is time-constructible, the inclusion $\mathrm{Aver}(\mathrm{NTIME}(t),\mathcal{F}) \subseteq \mathrm{Aver}(\mathrm{DSPACE}(O(t)),\mathcal{F})$ may not hold. This sheds light on the crucial difference between worst-case complexity and average-case complexity.

As an example, we shall demonstrate a closure property of average-case complexity class under disjoint union. It is easy to see in worst-case complexity theory that if A and B are in $\mathbf{P}$, then $A \oplus B$ is also in $\mathbf{P}$. In the average-case setting, we must consider the distributions as well as the sets of strings.

Lemma 3.5.16 *Let $\mathcal{C} \in \{\mathbf{P},\mathbf{NP},\mathbf{RP},\mathbf{BPP},\mathbf{PSPACE}\}$. Let $\mathcal{F}$ be a set of distributions which is closed under $\oplus$. If two distributional problems (A,μ_A) and (B,μ_B) are in $\mathrm{Aver}(\mathcal{C},\mathcal{F})$, then so is $(A \oplus B, \mu_A \oplus \mu_B)$.*

Proof. We prove the lemma for the case $\mathcal{C} = \mathbf{P}$. The other cases follow similarly. Assume that (A, μ_A) and (B, μ_B) are in $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$. There are two deterministic Turing machines M_A and M_B, and two polynomials p_A and p_B such that $L(M_A) = A$, $L(M_B) = B$, $\lambda x.\mathrm{Time}_{M_A}(x)$ is p_A on μ_A-average, and $\lambda x.\mathrm{Time}_{M_B}(x)$ is p_B on p_B-average. We assume without loss of generality that both p_A and p_B are increasing. Consider the following algorithm N.

begin simulation algorithm N
  input x
  if $x = \lambda$ **then** reject x
  find u such that either $x = 0u$ or $x = 1u$
  if $x = 0u$ **then** simulate M_A on input u
    else simulate M_B on input u
end.

Assume that $x = 0u$ for some u. It is easy to see that there is an absolute constant $c > 0$ which does not depend on the choice of x such that $\mathrm{Time}_N(x) \leq c \cdot (|x| + \mathrm{Time}_{M_A}(u) + 1)$. If we take a sufficiently large constant c, a similar inequality holds for M_B if $x \in 1\Sigma^*$.

To complete the proof, we show that $\lambda x.\mathrm{Time}_N(x)$ is polynomial on $\mu_A \oplus \mu_B$-average. First set $s(n) = c \cdot (n + p_A(n) + p_B(n) + 1)$. Note that s is a polynomial. For every $r \geq 1$,

$$\begin{aligned}
&\widehat{\mu_A \oplus \mu_B}(\{x \mid \mathrm{Time}_N(x) > s(|x| \cdot r)\}) \\
&= \frac{1}{2} \cdot \hat{\mu}_A(\{u \mid \mathrm{Time}_N(0u) > s(|0u| \cdot r)\}) + \frac{1}{2} \cdot \hat{\mu}_B(\{u \mid \mathrm{Time}_N(1u) > s(|1u| \cdot r)\}) \\
&\leq \frac{1}{2} \cdot \hat{\mu}_A(\{u \mid \mathrm{Time}_{M_A}(u) > p_A(|0u| \cdot r)\}) + \frac{1}{2} \cdot \hat{\mu}_B(\{u \mid \mathrm{Time}_{M_B}(u) > p_B(|1u| \cdot r)\}) \\
&\leq \frac{1}{2} \cdot \hat{\mu}_A(\{u \mid \mathrm{Time}_{M_A}(u) > p_A(|u| \cdot r)\}) + \frac{1}{2} \cdot \hat{\mu}_B(\{u \mid \mathrm{Time}_{M_B}(u) > p_B(|u| \cdot r)\}) \\
&< \frac{1}{2} \cdot \frac{1}{r} + \frac{1}{2} \cdot \frac{1}{r} = \frac{1}{r}.
\end{aligned}$$

Hence, N runs in polynomial time on ν-average. □

3.5.2 Inclusions and Separations

We shall discuss general separation and collapse results among average-case complexity classes introduced in the previous subsection.

The following proposition shows a basic separation between average-case complexity classes $\mathrm{Dist}(\mathcal{D}, \mathcal{F})$ and $\mathrm{Aver}(\mathcal{C}, *)$.

Let us recall from Chapter 2 that a tally set is a subset of $\{0\}^*$ and from Section 3.2 that ν_{tally} is the standard distribution that is positive only on $\{0\}^*$. We say a (worst-case) complexity class $\mathcal{C}$ is *closed under disjoint union* $\oplus$ if, for any sets A and B in $\mathcal{C}$, $A \oplus B$ is also in $\mathcal{C}$.

Proposition 3.5.17 *Let $\mathcal{C}$ and $\mathcal{D}$ be two complexity classes, and let $\mathcal{F}$ be a set of distributions with $\nu_{\mathrm{tally}} \in$*

$\mathcal{F}$. Assume that $\mathcal{D}$ contains the set $\{0\}^*$ and is closed under disjoint union. If $\mathbf{REC} - \mathcal{D} \neq \emptyset$ *and* $\mathrm{DTIME}(O(n)) \subseteq \mathcal{C}$, *then* $\mathrm{Aver}(\mathcal{C}, \mathcal{F}) \not\subseteq \mathrm{Dist}(\mathcal{D}, *)$.

Proof. Consider a set A in $\mathbf{REC} - \mathcal{D}$. Define $B = \{0\}^* \oplus A$. Thus, $(B, \nu_{\mathrm{tally}}) \notin \mathrm{Dist}(\mathcal{D}, *)$ since, otherwise, $A \in \mathcal{D}$ by the closure property of $\mathcal{D}$ under $\oplus$. We now show that $(B, \nu_{\mathrm{tally}}) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$. Let M be a deterministic Turing machine which computes A. This machine exists since A is recursive. Consider the following algorithm N for B:

begin simulation algorithm N for B
  input x
  if $x \in \{0\}^*$ **then** accept
  find y such that $x = 1y$
  simulate M on input y and **halt**
end.

Since $\mathrm{Time}_N(0^n) \leq cn + c$ for some absolute constant $c > 0$, we have

$$\hat{\nu}_{\mathrm{tally}}(\{x \mid \mathrm{Time}_N(x) > c|x| \cdot r + c\}) \leq \hat{\nu}_{\mathrm{tally}}(\{0^n \mid \mathrm{Time}_N(0^n) > cn + c\}) = \hat{\nu}_{\mathrm{tally}}(\emptyset) = 0 < \frac{1}{r}$$

for all $r \geq 1$. Hence, $(B, \nu_{\mathrm{tally}}) \in \mathrm{Aver}(\mathrm{DTIME}(cn + c), \mathcal{F}) \subseteq \mathrm{Aver}(\mathcal{C}, \mathcal{F})$. □

As a corollary, we get a result shown by Wang and Belanger [111] regarding the separation between $\mathrm{Dist}(\mathbf{NP}, *)$ and $\mathrm{Aver}(\mathbf{P}, *)$.

Corollary 3.5.18 **[111]** $\mathrm{Aver}(\mathbf{P}, \mathcal{F}) \not\subseteq \mathrm{Dist}(\mathbf{NP}, *)$ *for any set $\mathcal{F}$ of distributions with $\nu_{\mathrm{tally}} \in \mathcal{F}$.*

Next we shall introduce a notion of *weakly $\mathcal{C}$-descriptiveness* for average-case complexity classes $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$.

Definition 3.5.19 (Weakly Descriptiveness) Assume that $\mathcal{D}$ is a $\mathcal{T}$-time or $\mathcal{T}$-space bounded complexity class and $\mathcal{C}$ is any (worst-case) complexity class. We call the average complexity class $\mathrm{Aver}(\mathcal{D}, \mathcal{F})$ *weakly $\mathcal{C}$-descriptive* if, for every problem $(D, \mu) \in \mathrm{Aver}(\mathcal{D}, \mathcal{F})$, there exist a set $C \in \mathcal{C}$ and a deterministic oracle Turing machine M such that M with oracle C recognizes D in time $\mathcal{T}$ on μ-average.

Proposition 3.5.20 *Let $\mathcal{F}$ be a set of distributions.*

1. $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$ *is weakly* $\mathbf{P}$*-descriptive.*
2. $\mathrm{Aver}(\mathbf{NP} \cap \text{co-}\mathbf{NP}, \mathcal{F})$ *is weakly* $\mathbf{NP}$*-descriptive.*
3. $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$ *is weakly* $\mathbf{PP}$*-descriptive.*
4. $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$ *is weakly* $\mathbf{PSPACE}$*-descriptive.*

Proof. We shall show only the case (3). Let (D, μ) be a distributional problem in Aver($\mathbf{BPP}, \mathcal{F}$). Take a probabilistic Turing machine M which computes D in time polynomial on μ-average.

We define two sets C_0 and C_1 as follows: let C_0 be the set of strings $\langle 0, x, 1^n \rangle$ such that more than half of the computation paths of M on input x terminate in accepting configurations in less than n steps; and let C_1 be the set of strings $\langle 1, x, 1^n \rangle$ such that more than half the paths of M on x terminate in less than n steps. It is not hard to see that C_0 and C_1 are **PP** sets. Set $C = C_0 \cup C_1$.

Let us consider the following deterministic algorithm N with oracle C:

begin deterministic oracle Turing machine N with oracle C
  input x
  for $n = 1$ **to** ∞ **do**
    query $\langle 0, x, 1^n \rangle$ to oracle C
    if $\langle 0, x, 1^n \rangle \in C$ **then accept** and **halt**
    query $\langle 1, x, 1^n \rangle$ to oracle C
    if $\langle 1, x, 1^n \rangle \notin C$ **then reject** and **halt**
  end-for
end.

Recall the definition of $\mathrm{Time}^*_M(x)$ in Section 2.4. By our definition , the algorithm N with oracle C above repeats the **for**-loop $\mathrm{Time}^*_M(x) + 1$ times before accepting or rejecting x, because $\mathrm{Time}^*_M(x)$ is the minimal number of steps needed to check if M accepts or rejects x. Note that $\lambda x.\mathrm{Time}^*_M(x)$ is polynomial on μ-average due to Lemma 3.3.21. Furthermore, when the algorithm terminates, its output is always correct. Hence,

$$\mathrm{Time}^C_N(x) \leq d \cdot (\mathrm{Time}^*_M(x) + 1)$$

for some constant $d > 0$. Since $\lambda x.\mathrm{Time}^*_M(x)$ is polynomial on μ-average, obviously N with C computes D in polynomial time on μ-average. □

Provided that $\mathcal{C}$ is a $\mathcal{T}$-time or $\mathcal{T}$-space bounded complexity class, we say that Aver($\mathcal{C}, \mathcal{F}$) is *closed under weak description* if, for every $(C, \mu) \in \mathrm{Dist}(\mathcal{C}, \mathcal{F})$ and every deterministic oracle Turing machine M with oracle C which is $\mathcal{T}$-time bounded on μ-average, the distributional problem $(L(M, C), \mu)$ belongs to Aver($\mathcal{C}, \mathcal{F}$). For example, Aver($\mathbf{P}, \mathcal{F}$) and Aver($\mathbf{PSPACE}, \mathcal{F}$) are closed under weak description.

Lemma 3.5.21 *For $\mathcal{C} \in \{\mathbf{P}, \mathbf{PSPACE}\}$, Aver($\mathcal{C}, \mathcal{F}$) is closed under weak description.*

Proof. Here we shall show that Aver($\mathbf{PSPACE}, \mathcal{F}$) is closed under weak description. Let us first assume that (C, μ) is in Dist($\mathbf{PSPACE}, \mathcal{F}$). There exists a deterministic Turing machine M_0 computing C using polynomial space. Suppose $\mathrm{Space}_{M_0}(z) \leq p(|z|)$ for all z, where p is an appropriate increasing polynomial.

Consider any deterministic oracle Turing machine M which, with oracle C, is polynomial-time bounded on μ-average. We must construct a deterministic Turing machine which computes C using space polynomial

on μ-average without any help from oracles. The idea is to combine two Turing machines M_0 and M:

begin deterministic algorithm for N
  input x
  simulate M on input x
  while simulation **do**
    if M queries z **then** simulate M_0 on input z
    if M reaches a halting configuration **then output** $M(x)$ and **halt**
  end-while
end.

It is obvious that N computes C correctly without any oracles. The tape space used by N on input x, $\mathrm{Space}_N(x)$, is clearly bounded by

$$c \cdot \left(\mathrm{Time}_M^C(x) + \sum_{z \in Q(M,C,x)} \mathrm{Space}_{M_0}(z) + 1 \right).$$

To see that $\lambda x.\mathrm{Space}_N(x)$ is polynomial on μ-average, we must check $\lambda x. \sum_{z \in Q(M,C,x)} \mathrm{Space}_{M_0}(z)$ is polynomial on μ-average. This is seen as follows:

$$\begin{aligned} \sum_{z \in Q(M,C,x)} \mathrm{Space}_{M_0}(z) &\leq \max\{p(|z|) \mid z \in Q(M,C,x)\} \\ &\leq p(\mathrm{Time}_M^C(x)). \end{aligned}$$

Apply Lemma 3.3.14 to $p(\mathrm{Time}_M^C(x))$, and we conclude that $\lambda x.\mathrm{Space}_N(x)$ is polynomial on μ-average. This implies that $(L(M,C),\mu)$ belongs to $\mathrm{Aver}(\mathbf{PSPACE},\mathcal{F})$. □

Let us now see general separation and collapse results among average complexity classes. For the proposition below, we first introduce the supplementary notion of *safe* sets: given a set $\mathcal{T}$ of functions, we call $\mathcal{T}$ *safe* if (i) for any functions $t_1, t_2, t_3 \in \mathcal{T}$ and any constant $c \in \mathbb{N}$, there exists a function s in $\mathcal{T}$ such that $c(t_1(x) + t_2 \circ t_3(x) + 1) \leq s(x)$ for all x, and (ii) for each function $t \in \mathcal{T}$, $t(x) > t(y)$ implies $x > y$ for all x and y.

Proposition 3.5.22 *Let $\mathcal{C}$ be of the form* $\mathrm{DTIME}(\mathcal{T})$, $\mathrm{NTIME}(\mathcal{T})$, $\mathrm{RTIME}(\mathcal{T})$, $\mathrm{BPTIME}(\mathcal{T})$, $\mathrm{ATIME}(\mathcal{T})$, *or* $\mathrm{DSPACE}(\mathcal{T})$ *for some safe set $\mathcal{T}$ of functions on $\mathbb{R}^+$.*

1. *If $\mathcal{C} \subseteq \mathbf{REC}$, then $\mathcal{D} - \mathcal{C} \neq \emptyset$ implies* $\mathrm{Aver}(\mathcal{D},*) - \mathrm{Aver}(\mathcal{C},*) \neq \emptyset$.
2. *Assume that* $\mathrm{Aver}(\mathcal{D},\mathcal{F})$ *is weakly $\mathcal{D}'$-descriptive and* $\mathrm{Aver}(\mathcal{C},\mathcal{F})$ *is closed under weak description. Then, $\mathcal{D}' \subseteq \mathcal{C}$ implies* $\mathrm{Aver}(\mathcal{D},\mathcal{F}) \subseteq \mathrm{Aver}(\mathcal{C},\mathcal{F})$.

Proof. (1) The following proof is chiefly due to Karg and Schuler [48]. Assume that $D \in \mathcal{D} - \mathcal{C}$. Let us recall from Chapter 2 the notion of proper complexity cores. Since $\mathcal{C} \subseteq \mathbf{REC}$, by Lemma 2.5.12, there exists

a proper complexity core C for D with respect to $\mathcal{C}$. Note that $C \subseteq D$. Define the distribution μ as follows:

$$\hat{\mu}(x) = \begin{cases} \frac{1}{2n^2 \cdot \|C \cap \Sigma^n\|} & \text{if } x \in C \text{ and } |x| = n, \\ \frac{d}{2|x|^2} \cdot 2^{-|x|} & \text{otherwise} \end{cases}$$

for some adequate normalizing constant d. Since $D \in \mathcal{D}$, we have $(D, \mu) \in \mathrm{Dist}(\mathcal{D}, *) \subseteq \mathrm{Aver}(\mathcal{D}, *)$.

We next show that $(D, \mu) \notin \mathrm{Aver}(\mathcal{C}, *)$. Assume that $(D, \mu) \in \mathrm{Aver}(\mathcal{C}, *)$ via machine M. Suppose that $\mathcal{C}$ is a time-bounded complexity class and assume that $\lambda x.\mathrm{Time}_M(x)$ is t on μ-average for some $t \in \mathcal{T}$. Since C is a complexity core, there exists an integer $k > 0$ such that, for all $x \in C$ with $|x| \geq k$, $\mathrm{Time}_M(x) > t(2|x|^3)$ because, otherwise, the set $\{x \in C \mid \mathrm{Time}_M(x) < t(2|x|^3)\}$ is an infinite subset of C. Using this fact, we bound the probability of the event that $\mathrm{Time}_M(x) > t(|x| \cdot r)$ by

$$\begin{aligned} &\hat{\mu}(\{x \mid \mathrm{Time}_M(x) > t(|x| \cdot 2k^2)\}) \\ &\quad \geq \quad \hat{\mu}(\{x \in C \mid \mathrm{Time}_M(x) > t(|x| \cdot 2k^2)\}) \quad = \quad \hat{\mu}(C - \Sigma^{<k}) \\ &\quad = \quad \sum_{x \in C - \Sigma^{<k}} \frac{1}{|x|^2 \cdot \|C \cap \Sigma^{|x|}\|} \\ &\quad \geq \quad \frac{\|C \cap \Sigma^k\|}{2k^2 \cdot \|C \cap \Sigma^k\|} \quad = \quad \frac{1}{2k^2}. \end{aligned}$$

This contradicts the assumption $(D, \mu) \in \mathrm{Aver}(\mathcal{C}, *)$.

Note that if $\mathcal{C}$ is a space-bounded complexity class, use $\mathrm{Space}_M(x)$ instead of $\mathrm{Time}_M(x)$ in the above argument.

(2) Assume $\mathcal{D}' \subseteq \mathcal{C}$. Let (D, μ) be any distributional problem in $\mathrm{Aver}(\mathcal{D}, \mathcal{F})$. Since $\mathrm{Aver}(\mathcal{D}, \mathcal{F})$ is weakly $\mathcal{D}'$-descriptive, we can take a set D' in $\mathcal{D}$ and a deterministic oracle Turing machine M such that M is $\mathcal{T}$ on μ-average and computes D with oracle D'.

Notice that D' belongs to $\mathcal{C}$. The closure property of $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ under weak description leads to the conclusion that $(L(M, D')\mu)$ is in $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$. This implies the desired result that $(D, \mu) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$. □

As a corollary, we obtain the following result.

Corollary 3.5.23 *Let $\mathcal{F}$ be any set of distributions.*

1. $\mathbf{P} \neq \mathbf{BPP}$ *implies* $\mathrm{Aver}(\mathbf{P}, *) \neq \mathrm{Aver}(\mathbf{BPP}, *)$.
2. $\mathbf{P} = \mathbf{PP}$ *implies* $\mathrm{Aver}(\mathbf{P}, \mathcal{F}) = \mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$.

Interestingly, the class $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$ cannot be characterized by the notion of weak $\mathbf{NP}$-descriptiveness. Moreover, we are able to show that $\mathrm{Aver}(\mathbf{NP}, *)$ is indeed different from $\mathrm{Aver}(\mathbf{P}, *)$. The theorem below shows the separation between $\mathrm{Aver}(\mathbf{P}, *)$ and $\mathrm{Aver}(\mathbf{NP}, *)$ (see also [48]).

Theorem 3.5.24 **[97]** $\mathrm{Aver}(\mathbf{P}, *) \neq \mathrm{Aver}(\mathbf{NP}, *)$.

Proof. We shall construct a distributional decision problem (A, μ) which belongs to $\mathrm{Aver}(\mathbf{NP}, *)$ but

not to $\mathrm{Aver}(\mathbf{P}, *)$. First we choose a nonrecursive, recursively enumerable, tally set A whose elements are enumerated as $A = \{M(s_0), M(s_1), M(s_3), \ldots\}$ by some deterministic transducer M which always halts and outputs some strings. For simplicity, we assume without loss of generality that if M on input w produces x, then $\mathrm{Time}_M(w) \geq |x| + |w|$. We also choose the distribution μ defined as

$$\hat{\mu}(x) \propto \begin{cases} 2^{-3\mathrm{Time}_M(s_k)} & \text{if } x \in A, \text{ where } k = \min\{i \in \mathbb{N} \mid M(s_i) = x\}, \\ 2^{-2|x|} & \text{otherwise.} \end{cases}$$

Clearly the distribution μ is positive. We claim that the distributional problem (A, μ) belongs to $\mathrm{Aver}(\mathbf{NP}, *)$ but not to $\mathrm{Aver}(\mathbf{P}, *)$. We note that if $(B, \nu) \in \mathrm{Aver}(\mathbf{P}, *)$ and ν is positive, then B is recursive. Since A is not recursive, it is immediate that $(A, \mu) \notin \mathrm{Aver}(\mathbf{P}, *)$.

We next show that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, *)$. Let us consider the nondeterministic Turing machine N defined by the following simple nondeterministic algorithm. Remember that A is a tally set.

begin nondeterministic algorithm for N
  input x
  if $x \notin \{0\}^*$ **then reject**
  guess a natural number i (actually guess s_i)
  simulate M on input s_i
  if $x = M(s_i)$ **then accept else reject**
end.

Obviously N recognizes A. Now fix x and let $k = \min\{i \in \mathbb{N} \mid M(s_i) = x\}$. In the case where $x \notin A$, $\mathrm{Time}_N(x)$ is set to the shortest length of a rejecting path by our definition; thus, it is bounded by a constant independent of x. Let c_0 be such a constant. Hence,

$$\hat{\mu}(\{x \notin A \mid \mathrm{Time}_N(x) > c_0\}) = 0.$$

Suppose $x \in A$. Write D_x for the set $\{i \mid M(s_i) = x\}$. On input x, the above algorithm takes at most

$$\mathrm{Time}_N(x) \leq d \cdot (1 + |x| + \min\{|s_i| + \mathrm{Time}_M(s_i) \mid i \in D_x\}) \leq d \cdot (1 + |x| + |s_k| + \mathrm{Time}_M(s_k))$$

steps for some positive constant d which is independent of x. Since A is a tally set,

$$\hat{\mu}(\Sigma^n) = \hat{\mu}(x) = c \cdot 2^{-3\mathrm{Time}_M(s_k)},$$

where c is the normalizing constant for μ. We then have

$$\begin{aligned} \mathrm{Time}_N(x) &\leq d \cdot (2 \cdot \mathrm{Time}_M(s_k) + 1) \leq 2d \cdot 2^{\mathrm{Time}_M(s_k)} \\ &\leq \frac{2d}{|x|^2} \cdot 2^{3\mathrm{Time}_M(s_k)} \leq \frac{2cd}{|x|^2 \cdot \hat{\mu}(\Sigma^n)}. \end{aligned}$$

By Lemma 3.3.15, we conclude that $\lambda x.\mathrm{Time}_N(x)$ is polynomial on μ-average. □

Theorem 3.5.24 showed in the average-case setting a distinction between deterministic computation time and nondeterministic computation time. The distribution μ constructed in the proof of the theorem is

not effectively calculated and often takes extremely *small* probability. Nonetheless, most distributions we encounter in practice do not have any resemblance to this μ. When we discuss practical distributions, we may restrict our interest to such distributions that give relatively high probability for almost all input strings. Here we call such distributions *supportive*.

Definition 3.5.25 (Supportive Distributions) A distribution μ is called *supportive* if there exists a polynomial p such that either $\hat{\mu}(x) \geq 2^{-p(|x|)}$ or $\hat{\mu}(x) = 0$ for every x. A set of distributions is called *supportive* if all of its distributions are supportive. Let $SUPP$ be the collection of all supportive distributions.

For example, the standard distribution ν_{stand} is supportive since $\hat{\nu}_{\mathrm{stand}}(x) > \frac{1}{8(|x|+1)^2} \cdot 2^{-|x|} > 2^{-2|x|-5}$ for all x.

For any supportive set $\mathcal{F}$, the average complexity class $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$ is not a large class. The following proposition is a consequence of Lemma 3.3.11(2).

Proposition 3.5.26 *Let $\mathcal{F}$ be a set of distributions. If $\mathcal{F}$ is supportive, then* $\mathrm{Aver}(\mathbf{NP}, \mathcal{F}) \subseteq \mathrm{Dist}(\mathbf{NEXP}, \mathcal{F})$.

Proof. The result follows from the simple observation that, for a machine M which is t-time bounded on μ-average, $\mathrm{Time}_M(x) \leq t(|x|/\hat{\mu}(x)) \leq t(|x| \cdot 2^{p(|x|)})$ if $\hat{\mu}(x) \geq 2^{-p(|x|)}$. □

Regarding the $\mathrm{Dist}(\mathbf{NP}, \mathcal{F}) \subseteq ?\mathrm{Aver}(\mathbf{P}, *)$ question, it is enough to focus on supportive distributions in $\mathcal{F}$. We introduce an additional terminology: a set $\mathcal{F}$ of distributions is called *tame* if there exists a positive and supportive distribution.

Lemma 3.5.27 *Let $\mathcal{F}$ be a tame set of distributions and assume that $\mathcal{F}$ is closed under 2-addition. Then,* $\mathrm{Dist}(\mathbf{NP}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P}, *)$ *if and only if* $\mathrm{Dist}(\mathbf{NP}, SUPP \cap \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P}, *)$.

Proof. (Only if – part) Note that $\mathrm{Dist}(\mathbf{NP}, SUPP \cap \mathcal{F}) \subseteq \mathrm{Dist}(\mathbf{NP}, \mathcal{F})$. This inclusion obviously yields the claim.

(If – part) Let us assume that $\mathrm{Dist}(\mathbf{NP}, SUPP \cap \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P}, *)$. Moreover, we assume that (A, μ) is any distributional problem in $\mathrm{Dist}(\mathbf{NP}, \mathcal{F})$. We shall show that (A, μ) belongs to $\mathrm{Aver}(\mathbf{P}, *)$.

Since $SUPP \cap \mathcal{F} \neq \emptyset$, let us take a supportive distribution ν_0 from $\mathcal{F}$, and then define the default distribution ν as

$$\hat{\nu}(x) = \frac{1}{2} \cdot \hat{\nu}_0(x) + \frac{1}{2} \cdot \hat{\mu}(x).$$

By the assumption that $\mathcal{F}$ is closed under 2-addition, this distribution ν becomes a member of $\mathcal{F}$. It is also easy to see that ν is supportive, and consequently, we obtain $\nu \in SUPP \cap \mathcal{F}$.

From $\mathrm{Dist}(\mathbf{NP}, SUPP \cap \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P}, *)$, it follows that (A, ν) belongs to $\mathrm{Aver}(\mathbf{P}, *)$. Notice that ν p-dominates μ because $2 \cdot \hat{\nu}(x) \geq \hat{\mu}(x)$. Corollary 3.4.7 helps us conclude that (A, μ) is also in $\mathrm{Aver}(\mathbf{P}, *)$.

□

In the above lemma, we require $\mathcal{F}$ to be closed under 2-addition. Later we shall see several tame sets of distributions which contain the standard distribution and satisfy this requirement.

3.5.3 Another Characterization

Let us consider another characterization of basic average-case complexity classes, $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$, $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$, and $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$.

The following definition was proposed by Impagliazzo [43]. Let f be a function on Σ^* and S be a set. We say that an algorithm A *computes f with benign faults* if (i) it outputs either an element of $\mathrm{ran}(f)$ or "?" and (ii) on input x, if it outputs z which is not "?", then $z = f(x)$. An algorithm A computes S with benign faults if A computes χ_S with benign faults. We write $\mathrm{Time}_A(x, \delta)$ to denote the running time of $A(x, \delta)$.

Definition 3.5.28 (Benign Algorithm Scheme) [43] Let μ be a distribution and f be a function on Σ^*. A *polynomial-time benign algorithm scheme* for f on input ensemble $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ is an algorithm $A(x, \delta)$ such that

(i) there exists a polynomial p such that $\mathrm{Time}_A(x, \delta) \leq p(|x|, 1/\delta)$;

(ii) A computes f with benign faults; and

(iii) for all δ $(0 < \delta < 1)$ and all $n \in \mathbb{N}$, if $\mu_{\leq n}$ is defined, then $\hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid A(x, \delta) =?\}) < \delta$.

***Lemma 3.5.29* [43]** *Let $\mathcal{F}$ be a set of distributions. For a distributional problem (D, μ) with $\mu \in \mathcal{F}$, the following statements are equivalent:*

1. $(D, \mu) \in \mathrm{Aver}(\mathbf{P}, \mathcal{F})$.
2. *There is a polynomial-time deterministic benign algorithm scheme for D on $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$.*

Proof. First we shall see that (1) implies (2). Assume that (D, μ) is in $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$. There exists a deterministic Turing machine M such that $L(M) = D$ and Time_M is polynomial on μ-average. By Lemma 3.3.8, there is a polynomial p such that $\hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid \mathrm{Time}_M(x) > p(|x| \cdot r)\}) < \frac{1}{r}$ for all $n \in \mathbb{N}$ and all $r > 0$. We define the desired algorithm $A(x, \delta)$ as follows: on input (x, δ), A simulates M on input x for $p(|x|/\delta)$ steps and outputs "?" if M fails to halt. This algorithm is a benign algorithm for D on $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$ since

$$\begin{aligned} \hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid A(x, \delta) =?\}) &\leq \hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid \mathrm{Time}_M(x) > p(|x|/\delta)\}) \\ &< \frac{1}{1/\delta} = \delta. \end{aligned}$$

To see that (2) implies (1), we assume that $A(x, \delta)$ is a benign algorithm for D on input ensemble $\{\mu_{\leq n}\}_{n\in\mathbb{N}}$. We also assume that A runs in time $p(|x|, 1/\delta)$ for some fixed polynomial p. We may take a

unary polynomial q instead of p, because we can assume that p is increasing (i.e., $\lambda n.p(n,\delta)$ and $\lambda z.p(m,z)$ are increasing for each fixed δ and m); then setting $q(z) = p(z,z)$, we have $p(|x|, 1/\delta) \leq q(|x|/\delta)$. Moreover, we specify the form of $q(z)$ as $z^k + d$ by taking sufficiently large k and d. We then assume that the benign algorithm scheme $A(x,\delta)$ runs in time $(|x|/\delta)^k + d$.

Now let us consider the following algorithm M that computes D:

begin deterministic algorithm M for D
  input x
  for $i = 1$ to ∞
    simulate $A(x, 1/i)$
    if $A(x,1/i)$ outputs "?" **then** go to $(*)$
    output $A(x,1/i)$ and **halt**
$(*)$ **end-for**
end.

This algorithm M actually computes D. Let us define $s(z) = z^{k+1} + dz$ for all z. For any string x of length $\leq n$, if the algorithm M halts within the first r iterations of the **for**-loop, then

$$\mathrm{Time}_M(x) \leq \sum_{i=1}^{r}((i\cdot|x|)^k + d) \leq \left(\sum_{i=1}^{r} i^k\right)\cdot|x|^k + r\cdot d \leq (r|x|)^{k+1} + d\cdot r|x| = s(|x|\cdot r).$$

In other words, if $\mathrm{Time}_M(x) > s(|x|\cdot r)$, then $A(x,1/r) =?$. Hence, we have

$$\hat{\mu}_{\leq n}(\{x\in\Sigma^{\leq n} \mid \mathrm{Time}_M(x) > s(|x|\cdot r)\}) \leq \hat{\mu}_{\leq n}(\{x\in\Sigma^{\leq n} \mid A(x,1/r) =?\}) < \frac{1}{r}.$$

By Corollary 3.3.9, $\lambda x.\mathrm{Time}_M(x)$ is polynomial on μ-average. □

We observe that the quantifier characterization of nondeterministic and probabilistic Turing machines holds also in the average-case setting. Recall that, for instance, all sets in **NP** can be characterized in terms of an existential quantifier and sets in **P** as follows: a set A is in **NP** if and only if there exist a polynomial p and a set $B \in \mathbf{P}$ such that $A = \{x \mid \exists y[|y| \leq p(|x|) \wedge \langle x,y\rangle \in B]\}$ [118].

In the following, we shall give a logical characterization of the class $\mathrm{Aver}(\mathbf{NP},\mathcal{F})$.

Proposition 3.5.30 *Let $\mathcal{F}$ be a set of distributions. For every set A and every distribution μ in $\mathcal{F}$, the following statements are equivalent.*

1. $(A,\mu) \in \mathrm{Aver}(\mathbf{NP},\mathcal{F})$.
2. *There exists a function p from Σ^* to $\mathbb{N}$ and a set B in $\mathbf{P}$ such that*
   (i) p is polynomial on μ-average; and
   (ii) $A = \{x \mid \exists y[|y| \leq p(x) \wedge \langle x,y\rangle \in B]\}$.

Proof. The proof is straightforward and follows from the standard technique of encoding nondeterministic computation paths into strings and from the fact that $\lambda x.\mathrm{Time}_M(x)$ is polynomial on μ-average. □

First we need an amplification lemma.

Lemma 3.5.31 **(Amplification Lemma)** *Let μ be any distribution, A a set, and d a positive function computable in polynomial time on μ-average. Assume that a randomized Turing machine M satisfies $\mathbf{Pr}_M[M(x) = A(x)] \geq \frac{1}{2} + \frac{1}{d(x)}$ for all strings x. Then, there exists another randomized Turing machine N such that*

(i) $\lambda xys.\mathrm{Time}_N(x, y; s)$ is polynomial on $\mu \times \nu_{\mathrm{tally}}$-average; and

(ii) for each x and $m \in \mathbb{N}$, $\mathbf{Pr}_s[N(x, 1^m; s) = A(x) \mid s \in \Omega_N(x, 1^m)] \geq 1 - 2^{-|x|-m}$,

where $\mu \times \nu_{\mathrm{tally}}$ denotes the distribution μ' defined by $\hat{\mu}'(x, y) = \hat{\mu}(x) \cdot \hat{\nu}_{\mathrm{tally}}(y)$ for all pairs (x, y).

Proof. Let us assume that $\lambda xs.\mathrm{Time}_M(x; s)$ is t on μ-average for some increasing polynomial t. Assume also that d is computable in time polynomial on μ-average. For simplicity, we assume that $d(x) \geq 5$ for all x. We set $h(x, y) = \max\{1, |x| + |y|\}$ and let $p(x, y) = 2d(x)^3 h(x, y)$ for all pairs (x, y). First we show that p is polynomial on $\mu \times \nu_{\mathrm{tally}}$-average. For simplicity, we write μ' for $\mu \times \nu_{\mathrm{tally}}$ in the following argument.

Claim 1 *p is polynomial on μ'-average.*

Proof of Claim. Remember that d is polynomial on μ-average. Suppose that d is q on μ-average for some polynomial q, and define q' as $q'(z) = 2q(z)^3 z$. Obviously q' is a polynomial. Fix r arbitrarily such that $r \geq 1$. Let $E_r = \{(x, y) \mid d(x) > q(r \cdot |x|)\}$.

$$\begin{aligned}
&\hat{\mu}'(\{(x, y) \mid p(x, y) > q'(r \cdot (|x| + |y|))\}) \\
&\quad = \hat{\mu}'(\{(x, y) \mid d(x) > q(r \cdot |x|)\}) \\
&\qquad\qquad + \hat{\mu}'(\{(x, y) \in E_r \mid 2d(x)^3 h(x, y) > q'(r \cdot (|x| + |y|))\}) \\
&\quad \leq \frac{1}{r} + \hat{\mu}'(\{(x, y) \in E_r \mid |x| + |y| > r \cdot (|x| + |y|)\}) \\
&\quad = \frac{1}{r}.
\end{aligned}$$

Hence, p is polynomial on μ'-average. ■

Since p is polynomial on μ'-average, we can take a polynomial q such that p is q on μ'-average. The desired randomized machine N is defined by the following algorithm:

begin randomized algorithm for N
  input (x, y)
  if $y \notin \{1\}^*$ **then reject**

let $Acc := 0$ and $Rej := 0$
for $i = 1$ to $p(x,y)$ **do**
simulate M on input x
if M halts in an accepting state
then let $Acc := Acc + 1$ **else** let $Rej := Rej + 1$
end for
if $Acc \geq Rej$ **then accept else reject**
end.

Notice that the random-input domain for N is

$$\Gamma_N = \{(x,y,s) \mid \exists \{s_i\}_{1 \leq i \leq p(x,y)} [s = s_1 s_2 \cdots s_{p(x,y)} \wedge \forall i (1 \leq i \leq p(x,y) \rightarrow (x,s_i) \in \Gamma_M)]\}.$$

The running time of the machine N on input (x,y) with random input r, $\mathrm{Time}_N(x,y;s)$, is at most

$$c \cdot \left(\sum_{i=1}^{p(x,y)} \mathrm{Time}_M(x;s_i) + 1 \right)$$

for some absolute constant $c > 0$.

We first show that the random function $\lambda xys. \sum_{i=1}^{p(x,y)} \mathrm{Time}_M(x;s_i)$ is polynomial on μ'-average. To show this, we set $t_i(z) = (q(z) - i) \cdot t(z)$ for each number $i \in \mathbb{N}$, and we also set $t'(z) = c \cdot q(z) \cdot t(z) + 1$. Notice that $t'(z) = t_0(z)$. Let $r \geq 1$ be fixed. Let $D_r = \{(x,y,s) \in \Gamma_N \mid y \in \{1\}^* \wedge p(x,y) \leq q(r \cdot h(x,y))\}$. By our assumption, it is obvious that $\hat{\mu}'(\overline{D_r}) < \frac{1}{2r}$. It is enough to show that $\hat{\mu}'_{\Gamma_N}(\{(x,y,s) \in D_r \mid \mathrm{Time}_N(x,y;s) > t''(r \cdot (|x| + |y|))\}) < 1/r$ for some polynomial t''.

Let x be fixed and set $n = |x|$. For this x, write $P_{x,\ell}$ for $\mathbf{Pr}_s[\mathrm{Time}_M(x;s) > t(\ell \cdot n) \mid s \in \Gamma_M(x)]$. Also let y be 1^m. Set $t'(u,w) = c \cdot (t(u) \cdot q(w) + 1)$. Obviously t' is a polynomial.

Claim 2 *If $p(x,y) \leq q(r \cdot (n+m))$, then*

$$\mathbf{Pr}_s[\mathrm{Time}_N(x,y;s) > t'(\ell n, r(n+m)) \mid s \in \Gamma_N(x,y)] \leq q(r(n+m)) \cdot P_{x,\ell}.$$

Proof of Claim. The estimation is carried out as follows:

$$\begin{aligned}
&\mathbf{Pr}_s[\mathrm{Time}_N(x,y;s) > t'(\ell n, r(n+m)) \mid s \in \Gamma_N(x,y)] \\
&\quad \leq \mathrm{Pr}_s \left[\sum_{i=1}^{p(x,y)} \mathrm{Time}_M(x;s_i) > \sum_{i=1}^{p(x,y)} t(\ell n) \;\middle|\; s \in \Gamma_N(x,y) \right] \\
&\quad \leq \mathbf{Pr}_s[\mathrm{Time}_M(x;s_1) > t(\ell n) \mid s \in \Gamma_N(x,y)] \\
&\qquad\qquad + \mathrm{Pr}_s \left[\sum_{i=2}^{p(x,y)} \mathrm{Time}_M(x;s_i) > \sum_{i=2}^{p(x,y)} t(\ell n) \;\middle|\; s \in \Gamma_N(x,y) \right] \\
&\quad \leq P_{x,\ell} + \mathrm{Pr}_s \left[\sum_{i=2}^{p(x,y)} \mathrm{Time}_M(x;s_i) > \sum_{i=2}^{p(x,y)} t(\ell n) \;\middle|\; s \in \Gamma_N(x,y) \right]
\end{aligned}$$

Repeating this estimation, we reach the conclusion that

$$\mathbf{Pr}_s[\mathrm{Time}_N(x,y;s) > t'(\ell n, r(n+m)) \mid s \in \Gamma_N(x,y)] \leq \sum_{i=1}^{p(x,y)} P_{x,\ell} \leq q(r(n+m)) \cdot P_{x,\ell}.$$

The last inequality comes from the assumption that $p(x,y) \leq q(r(n+m))$. ■

We now define $t''(z) = c \cdot (t(4z^3 \cdot q(z)) \cdot q(z) + 1)$ for all z. Notice that

$$t''(r(n+m)) \geq t'(4n^3m^2r \cdot q(r(n+m)), r(n+m)) > t'(4n^3r \cdot q(r(n+m)), r(n+m)).$$

Hence, the rest of the calculation is carried out as follows. For any real number $r \geq 1$,

$$\begin{aligned}
&\hat{\mu}'_{\Gamma_N}(\{(x,y,s) \in D_r \mid \mathrm{Time}_N(x,y;s) > t''(r(|x|+|y|))\}) \\
&\quad = \hat{\mu}'_{\Gamma_N}(\{(x,1^m,s) \in D_r \mid \mathrm{Time}_N(x,1^m;s) > t'(|x| \cdot 4r|x|^2 \cdot q(r(|x|+m)), r(|x|+m))\}) \\
&\quad \leq \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\sum_{x:|x|=n} \hat{\mu}'(x,1^m) \\
&\qquad\qquad \times \mathbf{Pr}_s[p(x,1^m) \leq q(r(|x|+m)) \wedge \mathrm{Time}_N(x,1^m;s) > t'(|x| \cdot 4rn^2q(r(n+m)))] \\
&\quad \leq \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\sum_{x:|x|=n} \hat{\mu}(x) \cdot \hat{\nu}_{\mathrm{tally}}(1^m) \cdot q(r(n+m)) \cdot P_{x,4rn^2q(r(n+m))} \\
&\quad = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \hat{\nu}_{\mathrm{tally}}(1^m) \cdot q(r(n+m)) \cdot \hat{\mu}_{\Gamma_M}(\{(x,s) \mid \mathrm{Time}_M(x;s) > t(|x| \cdot 4rn^2q(r(n+m)))\}) \\
&\quad \leq \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{q(r(n+m)) \cdot \hat{\nu}_{\mathrm{tally}}(1^m)}{4r \cdot n^2 \cdot q(r(n+m))} = \frac{1}{4r}\sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{\hat{\nu}_{\mathrm{tally}}(1^m)}{n^2} \\
&\quad = \frac{\pi}{24r} < \frac{1}{2r}.
\end{aligned}$$

Therefore, $\lambda xys.\mathrm{Time}_N(x,y;s)$ is polynomial on $\mu \times \nu_{\mathrm{tally}}$-average.

Next we examine the error probability ρ_x that N on input x outputs a wrong answer; that is, $\mathbf{Pr}_N[N(x,1^m) \neq A(x)]$. Write τ_x for $\mathbf{Pr}_M[M(x) \neq A(x)]$, for comparison. The error probability ρ_x does not exceed the probability that M on input x does not compute $A(x)$ correctly in more than $\frac{p(x,y)}{2}$ $(= d(x)^3h(x,y))$ independent trials. Hence,

$$\rho_x \leq \sum_{k=0}^{p(x,y)/2} \binom{p(x,y)}{k} \tau_x^k \cdot (1-\tau_x)^{p(x,y)-k} \leq \left(\frac{p(x,y)}{2}+1\right) \binom{p(x,y)}{\frac{p(x,y)}{2}} \tau_x^{d(x)^2h(x,y)} \cdot (1-\tau_x)^{d(x)^3h(x,y)}.$$

Set $\epsilon_x = \frac{1}{2} - \tau_x \geq \frac{1}{d(x)}$. By Lemma A.9,

$$\begin{aligned}
\rho_x &\leq 2d(x)^3h(x,y) \cdot \frac{2^{p(x,y)+1}}{\sqrt{\pi p(x,y)}} \left(\frac{1}{2}+\epsilon_x\right)^{d(x)^3h(x,y)} \cdot \left(\frac{1}{2}-\epsilon_x\right)^{d(x)^3h(x,y)} \\
&= \frac{\sqrt{2d(x)^3h(x,y)}}{\sqrt{\pi}} \cdot 2^{p(x,y)+1} \cdot \left(\frac{1}{4}-\epsilon_x^2\right)^{d(x)^3h(x,y)} \\
&\leq \sqrt{d(x)^3h(x,y)} \cdot 2^{p(x,y)+1} \cdot 2^{-p(x,y)+1}(1-4\epsilon_x^2)^{d(x)^3h(x,y)}
\end{aligned}$$

$$\begin{aligned} &\leq 4 \cdot (d(x)h(x,y))^2 \cdot (1-4\epsilon_x^2)^{d(x)^3 h(x,y)} \\ &\leq e^{d(x)h(x,y)} \cdot \left(1 - \frac{4}{d(x)^3}\right)^{d(x)^3 h(x,y)} \\ &\leq e^{d(x)h(x,y)} \cdot e^{-4d(x)h(x,y)} = 2^{-3d(x)h(x,y)} < e^{-h(x,y)} \end{aligned}$$

since $(1-\frac{4}{n})^n \leq e^{-4}$ and $4n^2 \leq e^n$ for any number n greater than 4. Therefore, we obtain $\mathbf{Pr}_s[N(x,1^m;s) = A(x) \mid s \in \Gamma_N(x,1^m)] \geq 1 - 2^{-|x|-m}$. □

Corollary 3.5.32 *Let μ be a distribution and let d be a non-zero valued function computable in polynomial time on μ-average. Assume that, for a given set A, a randomized Turing machine M satisfies $\mathbf{Pr}_M[M(x) = A(x)] \geq \frac{1}{2} + \frac{1}{d(x)}$ for all strings x. For every function t which is computable in polynomial time on μ-average, there exists a randomized Turing machine N which runs in polynomial time on μ-average such that $\mathbf{Pr}_N[N(x) = A(x)] \geq 1 - 2^{-t(x)}$ for all strings x.*

Proof. Assume that M satisfies $\mathbf{Pr}_M[M(x) = A(x)] \geq \frac{1}{2} + \frac{1}{d(x)}$ for all x. Apply the Amplification Lemma, and we construct a randomized Turing machine N such that $\lambda xs.\mathrm{Time}_M(x,y,s)$ is polynomial on $\mu \times \nu_{\mathrm{tally}}$-average and $\mathbf{Pr}_s[N(x,1^m;s) = A(x)] \geq 1 - 2^{-|x|-m}$.

Given a function t computable in polynomial time on μ-average, we define another randomized Turing machine N' that takes input x and then simulates N on input $(x, 1^{t(x)})$. By the definition of N', the success probability $\mathbf{Pr}_N[N(x) = A(x)]$ is at least $1 - 2^{-|x|-t(x)} \geq 1 - 2^{-t(x)}$.

To complete the proof, it suffices to show that $\lambda xs.\mathrm{Time}_{N'}(x;s)$ is polynomial on μ-average. Notice that

$$\hat{\nu}_{\mathrm{tally}}(1^{t(x)}) = 2^{-2\mathrm{llog}(t(x))-1} \geq \frac{1}{8(t(x)+1)^2}$$

for all x. As a result, by Lemma 3.3.22, $\lambda xs.\mathrm{Time}_{N'}(x,s)$ is polynomial on μ-average. □

Proposition 3.5.33 *Let $\mathcal{F}$ be a set of distributions. For every set D and every distribution μ in $\mathcal{F}$, the following statements are equivalent:*

1. *$(D,\mu) \in \mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$.*
2. *For every function q that is computable in polynomial time on μ-average, there exists a probabilistic Turing machine M such that*
   - *(i) M is polynomial-time bounded on μ-average; and*
   - *(ii) $\mathbf{Pr}_M[M(x) = D(x)] \geq 1 - 2^{-q(x)}$ for all x.*
3. *There exist two functions h and d on Σ^*, and two randomized Turing machines (M_0, M_1) such that*
   - *(i) $\lambda x.|h(x)|$ is logarithmic on μ-average, and $h(x) \in \Gamma_{M_0}(x)$ for all x;*
   - *(ii) d is computable in polynomial time on μ-average;*

(iii) M_0 *and* $\lambda xs.M_1(\langle x, h(x)\rangle; s)$ *are polynomial-time bounded on* μ*-average; and*

(iv) for each $s' \in \Gamma_{M_0}(x)$, $\mathbf{Pr}_s[M_1(\langle x, M_0(x;s')\rangle; s) = D(x)] \geq \frac{1}{2} + \frac{1}{d(x)}$.

Proof. By applying Corollary 3.5.32 to (1), we immediately obtain (2). To see the implication from (2) to (3), set $q = \lambda x.2$ and take M satisfying conditions (i)-(ii) in (2). Let us define M_0 as the randomized Turing machine that works as follows: on input x, it outputs 0 without flipping any coins. Moreover, let M_1 works as follows: on input $\langle x, y\rangle$, simulate M on input x. Let $d(x) = 4$ and $h(x) = \lambda$ (the empty string). It easy to check conditions (i)-(iv) in (3).

Now it remains only to show that (3) implies (1). Assume that (3) holds. There are two randomized Turing machines (M_0, M_1) satisfying conditions (i)-(iv) in (3). By the Amplification Lemma, we can modify M_1 to another randomized Turing machine M_{amp} so that $\mathbf{Pr}_s[M_{amp}(x, 1^k, s'; s) = A(x, s')] \geq 1 - 2^{-k}$, where A is the set accepted by $M_1(\langle x, M_0(x;s')\rangle)$.

begin round robin algorithm for N
  input x (say, $n = |x|$)
  let $s := \lambda$ (empty string)
  for $k = 1$ **to** ∞ **do**
    for $j = 1$ **to** k **do**
      call subroutine $CHECK(x, s, j, k)$
    end for
  end for
end.

Here is the subroutine $CHECK(x, s, j, k)$:

begin subroutine $CHECK(x, s, j, k)$
  for all s' such that $|s'| = j$
    if either $M_0(x;s)$ does not halt or it halts
        without using up all random bits in s' **then** go to $(*)$
    (Assume that $M_0(x;s')$ halts after using up all random bits in s')
    let y be the output of $M_0(x;s)$
    compute $e := 2^j + 2$
    simulate the kth step of M_{amp} on input $(y, 1^e)$ using random seed s
    (If this step consists of a coin-flipping state, then flip a coin and let $s := sb$,
        where b is the outcome (0 or 1) of the coin toss.)
    if M_{amp} reaches a halting state **then** output $M_{amp}(\langle x, y\rangle, 1^e; s)$ and **halt**
$(*)$ **end for**
return.

Notice that random seed s is shared commonly by all computations of M_{amp} in the subroutine $CHECK$.

We shall perform worst-case analysis on the running time of the above algorithm. On input x along with random seed r, N takes steps at most

$$\mathrm{Time}_N(x;r) \leq c \cdot 2^{|h(x)|} \cdot (\mathrm{Time}_{M_1}(h(x);r') + 1)^2,$$

where r' is an initial segment of r. Notice that the function $\lambda x.2^{|h(x)|}$ is polynomial on μ-average by Lemma 3.3.6. Hence, $\lambda xr.\mathrm{Time}_N(x;r)$ turns out to be polynomial on μ-average.

Next we shall show that $\mathbf{Pr}_N[N(x) = A(x)] \geq \frac{1}{2} + \epsilon$. For each random seed, the success probability does not decline below 2/3. Thus, the probability that $N(x)$ coincides with $A(x)$ is at least 2/3. Therefore, (D, μ) belongs to $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$. □

We give a remark to the above proposition: if M_1 makes one-sided errors (i.e., $(D, \mu) \in \mathrm{Aver}(\mathbf{RP}, *)$), then we can replace the term $\frac{1}{2} + \frac{1}{d(x)}$ by $\frac{1}{d(x)}$.

3.6 Further Topics

This section discusses several topics which we have not covered in the previous sections.

Running Time of Nondeterministic Machines. Let us consider the running time of nondeterministic Turing machines. Wang and Belanger [111], Schuler and Yamakami [97], and Karg and Schuler [48] discussed a close connection between the $\mathbf{P} =?\mathbf{NP}$ question and the $\mathrm{Aver}(\mathbf{P}, \mathcal{F}) =?\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$ question based on the model of *clocked* nondeterministic Turing machines. As we have seen, this model guarantees that all computation paths in a computation tree of the clocked machines are of the same length.

Here we consider the model for which the running time of a nondeterministic Turing machine is defined to be the length of the shortest accepting computation paths when it accepts an input, or else the length of the longest rejecting computation paths. We call this restriction the *strict running time criterion* for the machine.

To avoid confusion, we use the notation $\mathrm{Aver}(\mathbf{NP}^*, \mathcal{F})$ to denote the average-case complexity class defined in terms of nondeterministic Turing machines, with the strict running time criteria, which run in polynomial time on average.

For this new class $\mathrm{Aver}(\mathbf{NP}^*, \mathcal{F})$, we are able to prove that $\mathrm{Aver}(\mathbf{NP}^*, \mathcal{F})$ is weakly **NP**-descriptive, and thus, $\mathbf{P} = \mathbf{NP}$ implies $\mathrm{Aver}(\mathbf{P}, \mathcal{F}) = \mathrm{Aver}(\mathbf{NP}^*, \mathcal{F})$ for any set $\mathcal{F}$ of distributions.

Claim 3 *The class* $\mathrm{Aver}(\mathbf{NP}^*, \mathcal{F})$ *is weakly* **NP***-descriptive.*

Proof of Claim. Let (D, μ) be a distributional problem in $\mathrm{Aver}(\mathbf{NP}^*, \mathcal{F})$. Let us consider a nondeterministic Turing machine M which computes D in polynomial time on μ-average.

Let C_0 be the set of strings of the form $\langle 0, x, 1^n \rangle$ such that M accepts x in less than n steps, and let C_1 be the set of strings of the form $\langle 1, x, 1^n \rangle$ such that there exist computation paths of M on input x which are

longer than or equal to n. Now write C for $C_0 \cup C_1$. Let us consider the following deterministic algorithm N with oracle C.

begin deterministic oracle Turing machine N with oracle C
  input x
  for $n = 1$ **to** ∞ **do**
    query $\langle 0, x, 1^n \rangle$ to oracle C
    if $\langle 0, x, 1^n \rangle \in C$ **then accept** and **halt**
    query $\langle 1, x, 1^n \rangle$ to oracle C
    if $\langle 1, x, 1^n \rangle \notin C$ **then reject** and **halt**
  end-for
end.

On any input x, this algorithm N with oracle C goes through the **for**-loop $\mathrm{Time}_M(x) + 1$ times until it terminates. Notice that when the algorithm terminates, its output is always correct. Hence,

$$\mathrm{Time}_N^C(x) \leq d \cdot (\mathrm{Time}_M(x) + 1)$$

for some constant $d > 0$. Since $\lambda x.\mathrm{Time}_M(x)$ is polynomial on μ-average, obviously N with C computes D in time polynomial on μ-average. ■

Heuristic Polynomial Time. Impagliazzo [43] introduced a new notion of "heuristic polynomial-time" class. We rephrase his definition in our setting. We define the heuristic polynomial-time class $\mathrm{Heur}(\mathbf{P}, \mathcal{F})$ as follows.

An *algorithm scheme* for (D, μ) is an algorithm $A(x, \delta)$ such that $\hat{\mu}_{\leq n}(\{x \in \Sigma^{\leq n} \mid A(x, \delta) \neq D(x)\}) \leq \delta$ for all $\delta > 0$ and all $n \in \mathbb{N}$. We say that a distributional decision problem (D, μ) is in $\mathrm{Heur}(\mathbf{P}, \mathcal{F})$ if there exists a polynomial-time deterministic algorithm scheme for (D, μ).

Obviously $\mathrm{Aver}(\mathbf{P}, \mathcal{F}) \subseteq \mathrm{Heur}(\mathbf{P}, \mathcal{F})$. It is an open problem whether $\mathrm{Dist}(\mathbf{NP}, \mathcal{F}) \subseteq \mathrm{Heur}(\mathbf{P}, \mathcal{F})$.

Chapter 4

Feasible Distributions

4.1 Introduction

Average-case analysis in general is sensitive to the choice of distributions, so that we need a careful study of the behaviors and properties of individual distributions. For example, to construct a better algorithm which runs *fast on average*, instances which occur with high probability should be solved quickly.

In statistics, Poisson distributions, for example, are commonly used for the analysis of events. This type of distribution is approximable by some appropriate polynomial-time algorithms.

In this chapter, we shall focus on *feasible distributions* which are either "feasibly computable" or "feasibly producible." This chapter will introduce two different categories of feasible distributions: *polynomial-time computable distributions* and *polynomial-time samplable distributions*. The former category was considered by Levin [60] and formulated by Gurevich [36]; the latter was introduced by Ben-David, Chor, Goldreich, and Luby [9] and has been studied in cryptography.

Following Ko and Friedman's [55] definition of *polynomial-time computable real functions*, Gurevich [36] took a similar step toward distributions. He called a distribution "computable in polynomial-time" if there is a polynomial-time approximation algorithm whose outputs asymptotically approach the values of a distribution within an exponentially small factor. Section 4.2 will discuss such distributions which are computable (or more accurately "approximable") in polynomial time by deterministic Turing machines.

In practice, rather than specifying *full* distributions, researchers often loosely define distributions by simply specifying associated semi-distributions. We note that all polynomial-time computable semi-distributions are effectively enumerated, while associated *full* distributions are not. Remember that our theory is based on *full* distributions, and therefore, whenever we use semi-distributions for the purpose of defining *full* distributions, we must guarantee the existence of such *full* distributions that are proportional to the semi-distributions almost everywhere. This process is called a *normalization* of semi-distributions. Unfortunately, not all computable semi-distributions are normalized to *full* distributions of the same complexity. Section 4.3 will show this negative result.

As Gurevich [36] pointed out, the reader may be cautious of the fact that the **P**-computability of a density function does not imply that of its associated distribution unless $\mathbf{P} = \mathbf{NP}$.

One of the most important results in Section 4.2 is the Distribution Controlling Lemma proven by Gurevich [36] and by Belanger and Wang [6]. This lemma enables us in Chapter 5 to prove the existence of complete distributional decision problems for Dist(**NP**, **P**-comp).

We shall turn our interest to instances which occur with low probability under *most* distributions computable in polynomial time. These instances are called *rare strings*. A set S is called *polynomial ℓ-rare* if, for any polynomial-time computable distribution μ, the set $\{x \in S \mid \hat{\mu}(x) > 2^{-\ell(|x|)}\}$ is finite.

Another category of feasible distributions is *samplable distributions* which were first introduced by Ben-David, Chor, Goldreich, and Luby [9] in their 1990 conference paper. Samplability, which is often found in statistical physics, is essentially a form of pseudo-randomness. For example, Ben-David *et al.* [9] showed that if pseudo-random generators exist, then polynomial-time samplable distributions are hard to approximate in polynomial time. This result will be extended in Section 4.7. In Section 4.4, sampling algorithms and samplable distributions will be defined.

Major Contributions. Most of the material in this chapter comes from Yamakami [119].

The formulation of **P**-samplable distributions given in this chapter is different from what has been defined by Ben-David, Chor, Goldreich, and Luby [9] and by Schuler and Watanabe [96], and therefore all the proofs are altered.

Given any **P**-computable distribution μ, Lemma 4.2.8 presents a sufficient condition of a function f so that the composition $\mu_{f^{-1}}$ becomes **P**-computable.

Theorem 4.2.14 shows the existence of polynomially ℓ-rare sets of distributions. This theorem relates to Kolmogorov complexity. The theorem actually proves that, for any increasing, unbounded function s, if ℓ satisfies $5\log n \leq \ell(n) \leq n$, then the set whose elements are not in $\mathrm{KT}[2\ell(n), 2^{\ell(n)+s(n)\cdot\log n}]$ is polynomially ℓ-rare.

Proposition 4.3.1 shows that there exists a **P**-computable semi-distribution which cannot be normalized by a standard method to a *full* distribution computable in polynomial time. In addition, Corollary 4.3.3 gives a sufficient condition for **P**-computable semi-distributions to be normalized by a standard method to *full* distributions of the same complexity.

Proposition 4.2.12 shows that there exists a positive **P**-computable distribution μ and an increasing, exp-honest function f in **FP** such that $\mu_{f^{-1}}$ is not p-dominated by any **P**-computable distribution.

In Proposition 4.4.7, **IP**-samp is shown to be included in **P**-samp.

Theorem 4.4.13 asserts that $\mathbf{P}^{\mathbf{BPP}}$-samp p-equal **P**-samp. This result is a counterpart of the result $\mathbf{BPP}^{\mathbf{BPP}} = \mathbf{BPP}$ in worst-case complexity theory.

One of the most important theorems in this chapter is Theorem 4.5.4, which shows that $\mathbf{P} = \mathbf{PP}$ if and only if **P**-comp = **P**-samp. Not all **P**-samplable distributions are therefore computable in polynomial time unless $\mathbf{P} = \mathbf{PP}$.

Proposition 4.6.3 shows that there is no p-universal distribution for **P**-comp if $\mathbf{P} = \mathbf{NP}$.

Theorem 4.7.3 shows that if every **P**-samplable distribution is p-dominated by some **P**-computable distribution, then every **NP** set is nearly-**RP**.

Corollary 4.7.7 shows #**P**-comp p-equals **P**-samp if **NP** $\subseteq$ **BPP**. This result follows Proposition 4.7.6, a careful modification of a result proven by Schuler and Watanabe [96].

Proposition 4.7.9 shows that $\mathbf{IP}_1$-samp is not avp-included in **P**-comp unless **P** = **RP**. Moreover, Theorem 4.7.12 shows that **P**-samp is not p-included in **P**-comp unless **P** = **NP**.

4.2 Computable Distributions

It seems natural to consider the sets of all feasibly "computable" distributions. Gurevich [36] first adapted the idea of the **P**-*computable real numbers* used by Ko and Friedman [55] to define polynomial-time computability of distributions. This section will further explore distributions *computable* in polynomial time.

4.2.1 Definition of Computable Distributions

Any distribution treated in this thesis is actually a real-valued function with Σ^* as its domain. Gurevich has applied the notion of *polynomial-time computability* of real-valued functions (introduced in Section 2.7) to distributions. Here we shall give a more general notion of *t-time computability* and *t-space computability* of semi-distributions for a function t, which is based on Gurevich's approximation scheme.

Let us recall how a deterministic Turing machine M equipped with an output tape computes a real number: the machine actually outputs a binary string w, which is interpreted as the dyadic rational number $\sum_{i=1}^{|w|} w_i \cdot 2^{-i}$, where $w = w_1 w_2 \cdots w_{|w|}$ and each w_i is a bit (in $\{0,1\}$). We say that M *approximates* real-value $\mu(x)$ if $|\mu(x) - M(x,0^i)| \leq 2^{-i}$ for all natural numbers i.

Definition 4.2.1 (Computable Distributions) [55, 36] Let t be a function on $\mathbb{N}$ and $\mathcal{T}$ be a set of functions on $\mathbb{N}$.

1. A semi-distribution μ is *recursive* (or *computable*) if there exists a deterministic Turing machine M equipped with two input tapes and one output tape, which *computes* μ; namely, on input $(x,0^i)$, $|\mu(x) - M(x,0^i)| \leq 2^{-i}$ for *all* strings $x \in \Sigma^*$ and *all* numbers $i \in \mathbb{N}$.

2. A semi-distribution μ is *t-time computable* (*t-space computable*, resp.) if there exists a deterministic Turing machine equipped with two input tapes and one output tape which, on input $(x,0^i)$, computes $\mu(x)$ in time (using space, resp.) $t(|x|,i)$.

3. A semi-distribution is *$\mathcal{T}$-time computable* (*$\mathcal{T}$-space computable*, resp.) if it is t-time (t-space, resp.) computable for some $t \in \mathcal{T}$.

Note that, by Lemma A.4, the Turing machine M in Definition 4.2.1 uniquely determines the distribution μ.

Figure 4.1 illustrates an asymptotic approach of the value $M(x,0^k)$ to the value of a distribution.

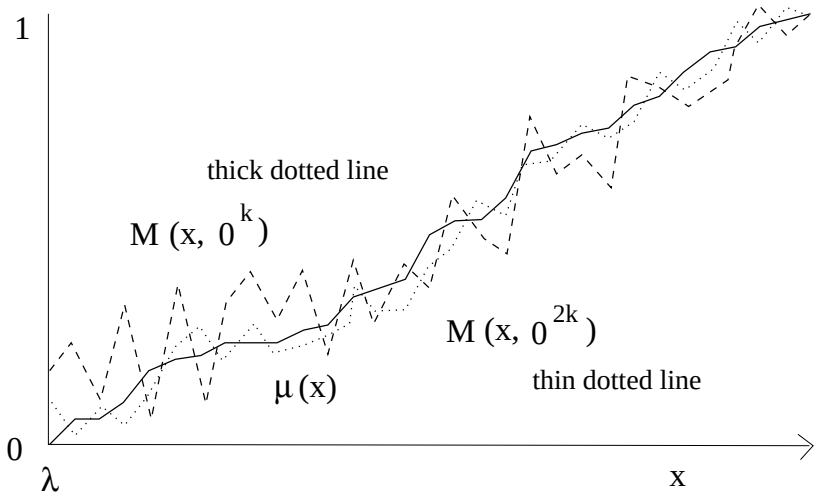

Figure 4.1: A distribution and its approximation

The reader should note that if we relax our definition of computability by requesting only the condition that $|\mu(x) - M(x, 0^i)| \leq 2^{-i}$ for "almost all" x and i, then semi-distributions are always normalized to *full* distributions of the same complexity. Here we mean by "normalization" the existence of the *full* distribution μ' satisfying that $\hat{\mu}'(x) = \hat{\mu}(x)$ for almost all strings x. (Another way to normalize a semi-distribution to a distribution is to multiply by a constant c so that $c \cdot \mu(x)$ converges to 1.) This subject will be discussed again in Section 4.3.

Note that, for each x, $\mu(x)$ is a t-time computable real number. Thus, if μ is t-time computable, then $\{\mu(x)\}_{x \in \Sigma^*}$ is an increasing sequence of t-time computable real numbers which converges to 1. But the converse may not be true in general.

We next introduce sets of distributions which are feasibly computable.

Definition 4.2.2 (Computable Distributions)

1. The notation **REC**-comp denotes the set of all recursive distributions.

2. A semi-distribution μ is called *polynomial-time computable* (**P**-computable, for short) if there exists a polynomial p such that μ is p-time computable. The notation **P**-comp denotes the collection of all **P**-computable distributions.

3. Similarly, a semi-distribution μ is called *linear-exponential-time computable* (**E**-computable, for short), *exponential-time computable* (**EXP**-computable, for short), *logarithmic-space computable* (**L**-computable), respectively, if μ is computable in linear-exponential time, in exponential time, and using logarithmic-space. We use the notations **E**-comp, **EXP**-comp, and **L**-comp, respectively, to denote the sets of

distributions which are **E**-computable, **EXP**-computable, and **L**-computable.

A simple example is the standard distribution ν_{stand}. It is easy to see from its definition that ν_{stand} is **L**-computable.

We can always assume that M on input $(x, 0^i)$ outputs at most $i+1$ bits. To see this, let M' be the machine that works as follows: it trims the binary fraction of $M(x, 0^{i+1})$ by rounding it down to $i+1$ bits; if the last bit is 1, then it adds 2^{-i-1} to make the number of bits at most $i+1$. This implies $|M(x, 0^i) - M'(x, 0^i)| \leq 2^{-i-1}$, and thus, we get

$$|\mu(x) - M'(x, 0^i)| \leq |\mu(x) - M(x, 0^{i+1})| + |M(x, 0^{i+1}) - M'(x, 0^i)| \leq 2^{-i-1} + 2^{-i-1} = 2^{-i}.$$

We shall exhibit several examples of **P**-computable distributions.

Example 4.2.3 The *uniform distribution* over a finite set of integers is **P**-computable, where we identify integers with strings. For two distinct natural numbers a and b, we define $\mu_{[a,b]}$ as follows:

$$\hat{\mu}_{[a,b]}(x) = \begin{cases} \frac{1}{a-b} & \text{if } s_a \leq x \leq s_b, \\ 0 & \text{otherwise.} \end{cases}$$

Example 4.2.4 Another common discrete distribution is the *Poisson distribution* with parameter $\lambda > 0$ that is defined on $\{0\}^*$ by

$$\hat{\mu}(1^n) = e^{-\lambda} \frac{\lambda^n}{n!}$$

for every number $n \in \mathbb{N}$. We here consider the case that λ is a positive rational number. Notice that the kth approximation of the value $\frac{\lambda^n}{n!}$ can be computed by a Turing machine in time polynomial in n and k. Similarly, by considering the power series for $e^{-\lambda}$, the number $e^{-\lambda}$ is **P**-computable. Hence, μ is **P**-computable.

Ben-David, Chor, Goldreich, and Luby [9], on the other hand, define polynomial-time computability of a distribution μ in such a way that, for all x, the value of $\mu(x)$ is exactly expressed in binary by some polynomial-time bounded transducer. To distinguish this concept from ours, let us call such distributions *strictly* **P***-computable* and denote by strict-**P**-comp the class of all strictly **P**-computable distributions. By this definition, if μ is strictly **P**-computable, then each value $\mu(x)$ is either 0 or at least $2^{-p(|x|)}$ for some polynomial p; thus, μ is supportive. This implies that the set strict-**P**-comp is supportive. Similarly, we use the notation strict-**REC**-comp to denote the set of all distributions which are *strictly computable* by deterministic Turing machines.

It is worth remarking that if a distribution is $\mathcal{T}$-time computable, then the density function is also $\mathcal{T}$-time computable. The converse, however, may not always hold since it is shown below that if $\mathbf{P} \neq \mathbf{NP}$, then there is a **P**-computable density function whose associated distribution cannot be computed in polynomial time. This result is due to Gurevich [36].

Lemma 4.2.5 **[36]** *Assuming* $\mathbf{P} \neq \mathbf{NP}$*, there exists a density function which is computable in polynomial time, but its associated distribution is not* $\mathbf{P}$*-computable.*

Proof. Assume that $\mathbf{P} \neq \mathbf{NP}$. Take a set A in $\mathbf{NP} - \mathbf{P}$ such that $A = \{x \mid \exists y[|y| = |x| \wedge \langle x, y\rangle \in B]\}$ for some set $B \in \mathbf{P}$.

Let $\hat{\nu}(x) = 0$ if $|x|$ is odd; otherwise, $\hat{\nu}(x) = \frac{2^{-2n}}{(n+1)(n+2)}$, where $|x| = 2n$. Clearly ν is in $\mathbf{P}$-comp. Now define the desired distribution μ as follows:

$$\hat{\mu}(z) = \begin{cases} \hat{\nu}(xy) & \text{if } z = x0y,\ |x| = |y|, \text{ and } \langle x, y\rangle \in B, \\ \hat{\nu}(xy) - \hat{\mu}(x0y) & \text{if } z = x1y \text{ and } |x| = |y|, \\ 0 & \text{otherwise.} \end{cases}$$

The following summation shows that this μ is truly a distribution:

$$\sum_z \hat{\mu}(z) = \sum_{(x,y):|x|=|y|} (\hat{\mu}(x0y) + \hat{\mu}(x1y)) = \sum_{(x,y):|x|=|y|} \hat{\nu}(xy) = 1.$$

Note that, by definition, $x \in A$ if and only if $\mu(x10^{|x|}) - \mu(x0^{|x|+1}) \neq 0$. If $\mu \in \mathbf{P}$-comp, then we have $A \in \mathbf{P}$, a contradiction. Therefore, μ is not in $\mathbf{P}$-comp. □

Another observation given by Gurevich [36] is:

Lemma 4.2.6 **[36]** *For every* $\mathbf{P}$*-computable distribution* μ*, there exists a polynomial-time deterministic Turing machine* M *such that:*

(i) $|\mu(x) - M(x, 0^i)| \leq 2^{-i}$ *for all* x *and* i*; and*

(ii) $\lambda x.M(x, 0^i)$ *is increasing (i.e.,* $x < y$ *implies* $M(x, 0^i) \leq M(y, 0^i)$*) for every* $i \in \mathbb{N}$*.*

Proof. Let μ be a distribution in $\mathbf{P}$-comp. By definition, there is a polynomial-time Turing machine N such that $|\mu(x) - N(x, 0^i)| \leq 2^{-i}$. We define the deterministic Turing machine M as follows:

begin deterministic algorithm for M
  input $(x, 0^k)$
  if $x = \lambda$ **then output** $N(\lambda, 0^k)$
  for all s $(0 \leq s \leq |x| + 1)$ **do**
    let $A(0^s) := \max\{N(0^r, 0^k) \mid r \leq s\}$
  end-for
  set $a := 0^{|x|}$ and $b := 0^{|x|+1}$
  repeat (binary search part)
    if $A(a) = A(b)$ **then output** $N(a, 0^k)$
    take the minimal c such that $\|\{z \mid a \leq z < c\}\| \geq \|\{z \mid c \leq z \leq b\}\|$
    if $N(c, 0^k) \leq A(a)$ **then** set $A(c) := A(a)$

else if $N(c, 0^k) > A(b)$ **then** set $A(c) := A(b)$
else set $A(c) := N(c, 0^k)$
if $x \leq c$ **then** set $b := c$ **else** set $a := c$
end-repeat
end.

To verify that M satisfies the required conditions is easy. □

Our approximation scheme is obviously different from the *floating point model*, another popular approximation scheme given by a *precision floating-point representation*, because, even if μ is **P**-computable, the kth bit of $\mu(x)$ may not be computable in time polynomial in $|x|$ and k. Nonetheless, the numbers represented in a finite precision model does not satisfy the law of associativity. Although we cannot know any exact precision of the value $\mu(x)$ in polynomial time in general, it is sufficient to know an approximate value $M(x, 0^i)$ within an exponential factor of its true value $\mu(x)$ when we consider the average behavior of algorithms under the distribution μ.

Part of the following useful observation was made by Gurevich [36] and by Wang and Belanger [112]. This lemma is the basis of the proofs of average **NP**-completeness shown in Chapter 5.

***Lemma 4.2.7* (Distribution Controlling Lemma) [36, 112]** *Let μ be a distribution in* **P**-comp.

1. *There exists a positive distribution ν in* strict-**P**-comp *such that, for every x, the value $\hat{\nu}(x)$ has at most $2|x| + 4$ binary digits, and $4 \cdot \hat{\nu}(x) > \hat{\mu}(x)$ holds.*
2. *There exists a total, one-one, p-invertible function $g \in \mathbf{FP}$ such that $\hat{\mu}(x) < 2^{-|g(x)|+2}$ for all x.*
3. *If μ is supportive, then there exists a total, one-one, p-invertible function $f \in \mathbf{FP}$ such that $4 \cdot 2^{-|f(x)|} \leq \hat{\mu}(x) \leq 20 \cdot 2^{-|f(x)|}$ for all x.*

Proof. (1) Assume $\mu \in$ **P**-comp. If there is an x such that $\mu(x) = 1$, then, since the set $\{y \mid y \leq x\}$ is finite, we can easily define ν which satisfies the claim. In the rest of the proof, we thus assume that $\mu(x) < 1$ for all x.

There exists a polynomial-time Turing machine M such that $|\mu(x) - M(x, 0^i)| < 2^{-i}$ for all x and i. For each string x, take $i = 2|x^+| + 1$ and let $N(x) = M(x, 0^{2|x^+|+1})$. Clearly we have $|\mu(x) - N(x)| < 2^{-2|x^+|-1}$. We trim the binary output of $N(x)$ by rounding it down to $2|x^+| + 1$ digits; if the last bit is 0, then cross it out; otherwise, add $2^{-2|x^+|-1}$ and then cross out the last bit in order to make the number of bits at most $2|x^+| + 1$. Now let $N'(x)$ be the result obtained by this process. If $x = \lambda$, then $N'(x)$ outputs the first two bits of the binary fraction of $N(x)$. Note that $|N(x) - N'(x)| \leq 2^{-2|x^+|}$. By the definition of N', we have

$$|\mu(x) - N'(x)| \leq |\mu(x) - N(x)| + |N(x) - N'(x)| < 2^{-2|x^+|-1} + 2^{-2|x^+|-1} = 2^{-2|x^+|}$$

for all $x \neq \lambda$. Hence, $|\mu(\lambda) - N'(\lambda)| < 1$ and $|\mu(x) - N'(x)| < 2^{-2|x^+|}$ for all nonempty strings x. Note that $N'(x)$ is expressed in binary with at most $2|x^+| + 2 \leq 2|x| + 4$ digits since $|x^+| \leq |x| + 1$.

To get the desired distribution, we define

$$\hat{\nu}(x) = \begin{cases} \frac{1}{4}(N'(\lambda)+1) & \text{if } x = \lambda, \\ \frac{1}{4}(N'(x) - N'(x^-) + 2^{-2|x|+1}) & \text{otherwise.} \end{cases}$$

Note that $\hat{\nu}(x) > 0$ for all x since $N'(x^-)$ has at most $2|x|$ bits. From our definition, it is easy to obtain that $4\nu(x) = 1 + N'(x) + 2\sum_{\lambda < z \leq x} 2^{-2|z|}$ for all nonempty strings z. Hence,

$$\lim_{x\to\infty} \nu(x) = \frac{1}{4}\left[1 + \lim_{x\to\infty} N'(x) + 2\sum_{n=1}^{\infty} 2^n \cdot 2^{-2n}\right] = 1.$$

For the empty string λ, $4\hat{\nu}(\lambda) = N'(\lambda) + 1 > \hat{\mu}(\lambda) - 1 + 1 = \hat{\mu}(\lambda)$. For the other strings x,

$$\begin{aligned} 4\hat{\nu}(x) &= N'(x) - N'(x^-) + 2^{-2|x|} \\ &> (\mu(x) - 2^{-2|x^+|}) - (\mu(x^-) + 2^{-2|x|}) + 2^{-2|x|+1} \\ &\geq \mu(x) - \mu(x^-) = \hat{\mu}(x). \end{aligned}$$

(2) Assume that $\mu \in \mathbf{P}$-comp and $\hat{\mu}(x) > 2^{-p(|x|)}$. Consider the distribution ν which is constructed in (1). Let $g(x) = \min\{y \mid \nu(x^-) < 0.y1 \leq \nu(x)\}$. This g is total, one-one, $\mathbf{P}$-computable, and p-invertible. Moreover, we have $\hat{\nu}(x) < 2^{-|g(x)|}$ for all x. This is seen as follows. By the minimality of $g(x)$, we have $0.w < \nu(x^-)$ and $\nu(x) \leq 0.w^+$, where $w = g(x)$. Hence, $\hat{\nu}(x) = \nu(x) - \nu(x^-) < 0.w^+ - 0.w = 2^{-|w|}$. Therefore, $\frac{\hat{\mu}(x)}{4} < 2^{-|g(x)|}$.

(3) Assume that $\hat{\mu}(x) > 2^{-p(|x|)}$ for some polynomial p. Take the function g in (2). Since $\lambda x.\frac{\hat{\mu}(x)}{4}$ is $\mathbf{P}$-computable, take a deterministic Turing machine M such that $|\frac{\hat{\mu}(x)}{4} - M(x, 0^i)| < 2^{-i}$. Since $\hat{\mu}(x) > 2^{-p(|x|)}$, we have $\frac{\hat{\mu}(x)}{4} > 2^{-p(|x|)-2}$. Let $N(x) = M(x, 0^{p(|x|)+4})$. Let $d(x)$ be the position of the leftmost digit 1 in the binary fraction of $N(x)$; that is, $2^{-d(x)} \leq N(x) < 2 \cdot 2^{-d(x)}$. Now define $f(x) = g(x)10^{d(x)-|g(x)|}$. Clearly f is a $\mathbf{P}$-computable, one-one, p-invertible function.

We next show that $5 \cdot 2^{-|f(x)|} > \frac{\hat{\mu}(x)}{4} \geq 2^{-|f(x)|}$. First we claim that $d(x) \leq p(|x|)+3$. Assume otherwise. By definition, $N(x) < 2^{-d(x)+1} \leq 2^{-p(|x|)-3}$, and then $\frac{\hat{\mu}(x)}{4} < N(x) + 2^{-p(|x|)-4} < 2^{-p(|x|)-2}$. This is a contradiction. Hence, we have

$$\frac{\hat{\mu}(x)}{4} < N(x) + 2^{-p(|x|)-4} \leq 2^{-d(x)-1} + 2^{-d(x)-1} = 5 \cdot 2^{-d(x)-1} = 5 \cdot 2^{-|f(x)|}.$$

To show that $\frac{\hat{\mu}(x)}{4} \geq 2^{-|f(x)|}$, let $d'(x)$ be the position of the leftmost digit 1 in the binary fraction of $\frac{\hat{\mu}(x)}{4}$. That is, $2^{-d'(x)} \leq \frac{\hat{\mu}(x)}{4} < 2 \cdot 2^{-d'(x)}$. We claim that $d'(x) \leq d(x)+1$. Assume otherwise. Since $N(x) \geq 2^{-d(x)}$ and $\frac{\hat{\mu}(x)}{4} < 2^{-d'(x)+1}$, we have $N(x) - \frac{\hat{\mu}(x)}{4} > 2^{-d(x)} - 2^{-d'(x)+1} \geq 2^{-d(x)-1}$. Then, $N(x) - \frac{\hat{\mu}(x)}{4} \geq 2^{-p(|x|)-4}$, a contradiction. Using the fact that $d'(x) \leq d(x)+1$, we conclude that $\frac{\hat{\mu}(x)}{4} \geq 2^{-d'(x)} \geq 2^{-d(x)-1} \geq 2^{-|f(x)|}$. □

The following lemma is easy to prove.

Lemma 4.2.8 *For any $\mathbf{P}$-computable distribution μ and any $\mathbf{P}$-computable function f on Σ^*, if f is increasing, then the default distribution $\mu_{f^{-1}}$ is also $\mathbf{P}$-computable.*

Proof. We assume that μ is a **P**-computable distribution witnessed by a deterministic polynomial-time Turing machine M. Assume also that a function f is **P**-computable.

Let us define another function g as $g(x) = \max\{z \mid f(z) \leq x\}$ for each x.

Claim 4 *The function g is* **P***-computable.*

Proof of Claim. Take an appropriate polynomial such that $|f(x)| \leq p(|x|)$ for all strings x. Using this upper bound of f, we compute g as follows:

begin deterministic algorithm for g
  input x
  compute the minimal n such that $|x| \leq p(n)$
  let $a := \lambda$ and let $b := 1^n$
  repeat (binary search part)
    if $a = b$ **then output** a
    take the minimal c such that $\|\{z \mid a \leq z < c\}\| \geq \|\{z \mid c \leq z \leq b\}\|$
    if $f(c) \leq x$ **then** set $a := c$ **else** set $b := c$
  end-repeat
end.

The binary search part of the above algorithm takes at most $O(n)$ steps, and as a result, the algorithm needs polynomially-many steps to compute $g(x)$. ■

Write ν for the distribution $\lambda x.\mu(g(x))$. We then have:

$$\begin{aligned}
\mu_{f^{-1}}(x) &= \sum_{z \leq x} \hat{\mu}_{f^{-1}}(z) = \sum_{z \leq x} \hat{\mu}(\{w \mid f(w) = z\}) \\
&\leq \hat{\mu}(\{w \mid f(z) \leq x\}) \leq \mu(\max\{w \mid f(w) \leq x\}) \\
&= \mu(g(x)) = \nu(x).
\end{aligned}$$

Let us define the deterministic Turing machine N as follows: on input $(x, 0^k)$, simulate M on input $(g(x), 0^k)$. Since g is **P**-computable, the machine N is also **P**-computable. To complete the proof, we must check the approximability of ν. This is seen as follows:

$$|\nu(x) - N(x, 0^k)| = |\mu(g(x)) - M(g(x), 0^k)| \leq 2^{-k}.$$

□

In complexity theory, a diagonalization argument is one of the most powerful and popular tools for discussing a separation of two complexity classes. Note that such an argument is based on the existence of an effective (i.e., recursive) enumeration of all elements in question. We already know that, for example, it is possible to enumerate all **P**-computable sets in an effective way.

The reader may wonder if one can effectively enumerate all distributions. Since **P**-computable distributions are characterized by Turing machines, we must enumerate Turing machines which compute such distributions. It seems, however, difficult to construct such an enumeration, because there are two objectives: (1) we must check whether the chosen Turing machine, say M, guarantees the convergence of its value to 1 (i.e., $\lim_{x\to\infty} M(x,0^i)=1$) (2) we must effectively check whether the machine satisfies the convergence scheme, $|M(x,0^i)-M(x,0^j)|\leq 2^{-i}+2^{-j}$.

To avoid checking the convergence scheme, let us turn our attention to strictly **P**-computable distributions. By Lemma 4.2.7(1), we can bound an arbitrary **P**-computable distribution by another strictly **P**-computable distribution with a constant factor. For most applications, we therefore consider strictly **P**-computable distributions. Can we now enumerate all strictly **P**-computable distributions ? We still have to resolve the problem of checking the convergence of a machine. Actually we do not need to restrict attention to *full* distributions. Schuler [92] first pointed out the existence of an effective enumeration of all strictly **P**-computable semi-distributions.

Theorem 4.2.9 **[92]** *There exists an effective enumeration of all strictly* **P***-computable semi-distributions.*

Proof. The method of proving this lemma is basically the same as in Lemma 4.2.6. Take any effective enumeration of polynomial-time deterministic Turing machines, say $\{M_i\}_{i\in\mathbb{N}}$. We modify each machine M_i into another machine M_i' which is increasing by an algorithm similar to that presented in the proof of Lemma 4.2.6 (by replacing N by M_i and M by M_i'). The sequence $\{M_i'\}_{i\in\mathbb{N}}$ obtained by this modification also becomes an effective enumeration of deterministic Turing machines, and each M_i' computes some semi-distribution in polynomial time. □

From the above theorem, we remark that we cannot exclude the trivial semi-distribution from the enumeration.

4.2.2 Rare Strings and Rare Sets

This section will consider input strings which occur rarely for most **P**-computable distributions. These strings need special attention because the set of such strings consists of the most difficult instances for most algorithms to work on in polynomial time.

Definition 4.2.10 (Rare Strings) Let $\mathcal{F}$ be an enumeration of semi-distributions, say $\mathcal{F}=\{\mu_0,\mu_1,\ldots\}$, and let k,s be functions from $\mathbb{N}$ to $\mathbb{R}^+$. A string x is *rare with respect to* $(k,s,\mathcal{F})$ if $\hat{\mu}_i(x)\leq 2^{-s(|x|)}$ holds for all $i<k(|x|)$. Let $\mathrm{RARE}(k,s,\mathcal{F})$ be the collection of all rare strings with respect to $(k,s,\mathcal{F})$.

Note that the rareness of a string depends on the enumeration $\mathcal{F}$.

Lemma 4.2.11 *Let $\mathcal{F}$ be an enumeration of all* **P***-computable semi-distributions. Let k be any increasing function on $\mathbb{N}$ such that $0 < k(n) \leq \frac{n}{9}$ for all n. Assume $s(n) < n + \log n - 2\log k(9n)$. For all $n_0 > 0$, there exists an n with $n_0 \leq n \leq 9n_0$ such that $\|\mathrm{RARE}(k, s, \mathcal{F}) \cap \Sigma^n\| \geq 2^n - 2^{s(n)-\log n+2\log k(9n)}$.*

Proof. We first show that, for any integer $n_0 > 0$, there exists an n with $n_0 \leq n \leq 3n_0 + 6k(n_0)$ such that, for each $i < k(n_0)$, $\|\{x \in \Sigma^n \mid \hat{\mu}_i(x) > 2^{-s(n)}\}\| < \frac{k(n)\cdot 2^{s(n)}}{n}$. Assume otherwise. Let $r(n) = 3n + 6k(n)$ and define $A_n^i = \{x \in \Sigma^n \mid \hat{\mu}_i(x) > 2^{-s(n)}\}$. Take n_0 such that, for all n with $n_0 \leq n \leq r(n_0)$, there exists an $i < k(n_0)$ satisfying $\|A_n^i\| \geq k(n) \cdot 2^{s(n)}/n$. Hence, at least $\lfloor \frac{r(n_0)-n_0+1}{k(n_0)} \rfloor$ many n's satisfy the condition $\|A_n^j\| \geq k(n) \cdot 2^{s(n)}/n$ for some $j < k(n_0)$. Let $c = \frac{(k(n_0)-1)(r(n_0)+1)+n_0}{k(n_0)}$. Since $r(n_0) \geq \frac{n_0+2k(n_0)-1}{e^{-1/k(n_0)}}$, we have $\left(\frac{r(n_0)}{c+1}\right)^{k(n_0)} \geq e$. Then,

$$\sum_x \hat{\mu}_j(x) \geq \sum_{n=\lceil c \rceil}^{r(n_0)} \sum_{x \in A_n^i} \hat{\mu}_i(x) \geq \sum_{n=\lceil c \rceil}^{r(n_0)} \frac{k(n)\cdot 2^{s(n)}}{n} \cdot 2^{-s(n)} = \sum_{n=\lceil c \rceil}^{r(n_0)} \frac{k(n)}{n} > \int_{c+1}^{r(n_0)} \frac{k(n)}{x} dx \geq 1.$$

The lemma immediately follows from the following inequality:

$$\begin{aligned} \|\{x \in \Sigma^n \mid \exists i < k(n)[\hat{\mu}_i(x) > 2^{-s(n)}]\}\| &\leq \sum_{i<k(n)} \|\{x \in \Sigma^n \mid \hat{\mu}_i(x) > 2^{-s(n)}\}\| \\ &< k(n) \cdot \frac{k(9n)\cdot 2^{s(n)}}{n} \leq 2^{s(n)-\log n+2\log k(9n)}. \end{aligned}$$

□

We can construct a **P**-computable distribution μ and an exp-honest, **P**-computable function f such that no distributions in **P**-comp p-dominate $\mu_{f^{-1}}$. Recall from Chapter 2 that μ *p-dominates* ν if and only if there is a p-bounded function p from Σ^* to $\mathbb{R}^+$ such that $p(x) \cdot \hat{\mu}(x) \geq \hat{\nu}(x)$ for all x.

Proposition 4.2.12 **[119]** *There exists a positive* **P***-computable distribution μ and an increasing, exp-honest function f in* **FP** *such that $\mu_{f^{-1}}$ is not p-dominated by any* **P***-computable distributions.*

Proof. We first define η as follows:

$$\hat{\eta}(x) = \begin{cases} 2^{-2\mathrm{llog}(n-1)-1} & \text{if } x \in \{0\}^* \text{ and } |x| = n^6 \text{ for some } n \geq 2, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\hat{\mu}(x) = \frac{1}{2}\hat{\nu}_{\mathrm{stand}}(x) + \frac{1}{2}\hat{\eta}(x)$. The distribution μ is positive and obviously **P**-computable. For every $n \geq 2$ and for $x = 0^{n^6}$, we have

$$\hat{\mu}(x) > \frac{1}{2}\hat{\eta}(x) = \frac{1}{2\cdot 2^{2\mathrm{llog}(n-1)}} \geq \frac{1}{2n^2} \geq \frac{1}{n^3} = \frac{1}{(n^6)^{1/2}} = \frac{1}{|x|^{1/2}}$$

since $2^{\mathrm{llog}(n-1)} \leq n$. Hence, $\hat{\mu}(x) > 1/|x|^{1/2}$ holds for all $x = 0^{n^6}$ with $n \geq 2$.

To define the desired function f, we need an effective enumeration of all strictly **P**-computable semi-distributions. Let $\mathcal{F} = \{\nu_i \mid i \in \mathbb{N}\}$ be such an enumeration by Theorem 4.2.9. Let $f(x)$ be the minimum

y such that $\log n \leq |y| \leq 9 \log n$ and $|y|^{k-1} \cdot \hat{\nu}_i(y) < \hat{\mu}(0^n)$ for all $i < \log n$ and all integers k with $1 \leq k \leq \frac{\log n}{5+2 \log \log n}$, where $n = \min\{r \mid r^6 \leq |x| < (r+1)^6\}$ if $n \geq 2^{13}$; otherwise, let $f(x) = x$.

Function f is well-defined. To see this, consider the case $|x| = n^6$ for some $n \geq 2^{13}$. By choosing $\log n$ as $k(n)$ and $n + \log n - 2 \log \log n - 4$ as $s(n)$ in Lemma 4.2.11, we know that there exists at least one rare string y with respect to $(k, s, \mathcal{F})$ with $\log n \leq |y| \leq 9 \log n$, i.e., $\hat{\nu}_i(y) \leq 2^{-s(n)} = \frac{16 \log^2 |y|}{|y| \cdot 2^{|y|}}$ for all $i < \log n$. For such a string y, we have

$$|y|^{k-1} \cdot \hat{\nu}_i(y) \leq \frac{16 |y|^{k-2} \log^2 |y|}{2^{|y|}} \leq \frac{16 \cdot 9^{k-1} \log^{k-1} n \cdot \log^2 (9 \log n)}{n} \leq \frac{9^k \log^k n}{n} \leq \frac{1}{n^{1/2}} < \hat{\mu}(0^n)$$

since $\log n \leq |y| \leq 9 \log n$, $16 \log^2 (9 \log n) \leq (9 \log n)^2$ if $n \geq 4$, and $9^k \log^k n \leq \sqrt{n}$ if $k \leq \frac{\log n}{5+2 \log \log n}$. Hence, $f(x)$ exists. It is easy to see that f is exp-honest and also polynomial-time computable.

By definition, for all k and i, $|y|^{k-1} \cdot \hat{\nu}_i(y) < \hat{\mu}_{f^{-1}}(y)$ for some y, since $\hat{\mu}_{f^{-1}}(y) \geq \hat{\mu}(0^n)$. □

Inspired by the notion of complexity cores, we shall introduce sets which consist of instances which occur rarely under most distributions in question.

Definition 4.2.13 (Polynomially ℓ-Rare Sets) Let ℓ be a function on $\mathbb{N}$ and let $\mathcal{F}$ be a set of distributions. A set S is called *ℓ-rare with respect to $\mathcal{F}$* if, for any distribution μ in $\mathcal{F}$, the set $\{x \in S \mid \hat{\mu}(x) > 2^{-\ell(|x|)}\}$ is finite. If $\mathcal{F}$ is **P**-comp, then we simply call this set S *polynomially ℓ-rare*.

In what follows, we shall show the existence of a polynomially ℓ-rare set for increasing, unbounded l with the extra condition $5 \log n \leq \ell(n) \leq n$. The proof uses Kolmogorov complexity.

Theorem 4.2.14 *Let ℓ be any function on $\mathbb{N}$ which satisfies $5 \log n \leq \ell(n) \leq n$ for all $n \in \mathbb{N}$. For any increasing, unbounded function s on $\mathbb{N}$, the set $\{x \mid x \notin \mathrm{KT}[2\ell(n), 2^{\ell(n)+s(n) \cdot \log n}]\}$ is polynomially ℓ-rare.*

Proof. Assume that s is increasing and unbounded. In what follows, we show a general statement: for every distribution $\mu \in$ **P**-comp, there exists a positive integer n_0 such that, for all strings x of length $\geq n_0$, if $x \notin \mathrm{KT}[2\ell(n), 2^{\ell(n)+s(n) \log n}]$, then $\hat{\mu}(x) \leq 2^{-\ell(|x|)}$. This clearly implies the theorem.

For a distribution $\mu \in$ **P**-comp, using Lemma 4.2.6, we can take a deterministic polynomial-time Turing machine M such that, for all $i \in \mathbb{N}$, $|\mu(x) - M(x, 0^i)| \leq 2^{-i}$ and $\lambda x.M(x, 0^i)$ is increasing. We let $T(\lambda) = M(\lambda, 0)$ and $T(x) = M(x, 0^{|x|+3}) - M(x^-, 0^{|x|+3})$ for $x \in \Sigma^+$. For any set A, write $T(A)$ for $\sum_{x \in A} T(x)$. Notice that T may not be a distribution, but $T(x) \geq 0$ holds for all x. Moreover, T approximates $\hat{\mu}$ since it follows that

$$\begin{aligned} |\hat{\mu}(x) - T(x)| &\leq |\mu(x) - M(x, 0^{|x|+3})| + |\mu(x) - M(x, 0^{|x|+3})| \\ &\leq 2^{-n-3} + 2^{-n-3} = 2^{-n-2}. \end{aligned}$$

From the fact that T is **P**-computable, we now suppose that T on input x is computable in $|x|^d + d$ steps for some positive integer d independent of x.

Consider an integer n_0 large enough so that the following inequalities always hold: $\log n_0 \geq 2\log i_0 + 9$, $n_0 \geq 4c$, and $n_0 \geq d + 4$, where i_0 and c are constants given later (not depending on the choice of n_0). For any integer n not smaller than n_0, let

$$A_n = \{x \in \Sigma^n \mid T(x) + 2^{-n-2} > 2^{-\ell(n)}\}.$$

As for the cardinality, we can claim that $\|A_n\| < 2^{\ell(n)+1}$.

Claim 5 *$\|A_n\| < 2^{\ell(n)+1}$ for all integers $n \geq n_0$.*

Proof of Claim. Assume otherwise. Note that, for each $x \in A_n$, $T(x) > 2^{-\ell(n)} - 2^{-n-2}$. Hence, we have

$$\begin{aligned}\sum_{x\in A_n} \hat{\mu}(x) &> \sum_{x\in A_n} (T(x) - 2^{-n-2}) > \|A_n\| \cdot (2^{-\ell(n)} - 2\cdot 2^{-n-2}) \\ &> 2^{\ell(n)+1} \cdot (2^{-\ell(n)} - 2^{-n-1}) \\ &= 2 - 2^{\ell(n)-n}\end{aligned}$$

Since $\ell(n) \leq n$, $\sum_{x\in A_n} \hat{\mu}(x) > 2 - 1 = 1$. This is obviously a contradiction. Therefore, $\|A_n\| < 2^{\ell(n)+1}$. ■

We note that, for each $x \in \Sigma^n$, if $\hat{\mu}(x) \leq 2^{-\ell(n)}$, then $x \in A_n$. To get the desired consequence, we shall show that $A_n \subseteq \mathrm{KT}[2\ell(n), 2^{\ell(n)+s(n)\log n}]$. Fix an integer e with $1 \leq e < 2^{\ell(n)+1}$. Let us take the eth element of A_n in the standard order on Σ^* and consider the following deterministic algorithm:

begin deterministic algorithm
  input $\langle s_l, s_n, s_e\rangle$
  let $j := 0$
  for all strings y in Σ^l **do**
    compute the value $T(y)$ (let this term to be t)
    if $t + 2^{-n-2} \leq 2^{-l}$ **then** go to $(*)$
    let $j := j + 1$
    if $j = e$ **then output** y
$(*)$ **end-for**
  output 0
end.

Let i_0 be an index of the Turing machine that runs the above algorithm. Note that the description of this algorithm needs only the index of T.

Recall the definition of the universal Turing machine U in Section 2.7. On each input $\langle s_{i_0}, s_{\ell(n)}, s_n, s_e\rangle$, the machine U outputs w_e. Let $n \geq n_0$. From the fact that $|\langle x_1, x_2, x_3, x_4\rangle| \leq \sum_{j=1}^{3} |x_j| + |x_4| + 1$, it follows that, for $k \in \{i_0, \ell(n), n, e\}$, $|s_k| = \mathrm{llog}(k) \leq 1 + \log k$. In particular,

$$|s_e| \leq 1 + \log e \leq 1 + (\ell(n) + 1) = \ell(n) + 2.$$

Thus, the size of input string $\langle s_{i_0}, s_{\ell(n)}, s_n, s_e \rangle$ is bounded above by

$$\begin{aligned} & 2|s_{i_0}| + 2|s_{\ell(n)}| + 2|s_n| + |s_e| + 1 \\ & \quad \leq \quad 2\log i_0 + 2\log \ell(n) + 2\log n + \ell(n) + 9 \\ & \quad \leq \quad 3\log n + 2\log \ell(n) + \ell(n) \quad \leq \quad 5\log n + \ell(n) \\ & \quad \leq \quad \ell(n) + \ell(n) \quad = \quad 2\ell(n) \end{aligned}$$

since $\log n \geq 2\log i_0 + 10$ and $5\log n \leq \ell(n) \leq n$ for all integers $n \geq n_0$.

Finally, we shall analyze the running time of U. Notice that, in the original algorithm as defined above, there are 2^l iterations of the **for**-loop, and on each of such iterations, for $y \in \Sigma^l$, the algorithm needs steps as many as $O(\mathrm{Time}_T(y) + \log n + l + \log e + 1)$. Therefore, the running time of U on input $\langle s_{i_0}, s_{\ell(n)}, s_n, s_e \rangle$ is at most, for some constant $c > 0$ (not depending on the choice of n),

$$\begin{aligned} & c \cdot 2^{\ell(n)} \left((n^d + d) + n + \ell(n) + (\ell(n) + 2) \right) \\ & \quad \leq \quad c \cdot 2^{\ell(n)} \cdot (2\ell(n) + 2n^d + d + 2) \quad \leq \quad 2^{\ell(n)} \cdot c \cdot (2n + 3n^d) \\ & \quad \leq \quad 2^{\ell(n)} \cdot 5c \cdot n^d \quad \leq \quad 2^{\ell(n)} \cdot n^{d+1} \\ & \quad \leq \quad 2^{\ell(n)} \cdot 2^{(d+1)\log n} \quad \leq \quad 2^{\ell(n) + s(n)\log n} \end{aligned}$$

since $5c \leq n$, $\ell(n) \leq n$, and $d + 2 \leq n^d$. Therefore, we conclude that $w_e \in \mathrm{KT}[2\ell(n), 2^{\ell(n)+s(n)\log n}]$. This completes the proof. □

Note that, by Lemma 2.7.7, the set we have used in the above theorem belongs to the class $\mathrm{DTIME}(O(2^{3\ell(n)+s(n)\log n}))$.

The following lemma of Schuler [93] has the same flavor as Theorem 4.2.14 and will be used in Section 5.6. Let k, s and ℓ be functions on $\mathbb{N}$. Assume that ℓ and s are unbounded and increasing. Moreover, assume that $k(n) \cdot \ell(n) \leq n$, $\ell(n) \geq 10\log n$, and $k(n) \geq 1$ for almost all n. For each integer $n > 0$, we define $k(n)$ strings $z_1^n, \ldots, z_{k(n)}^n$ as follows:

$$z_i^n = \begin{cases} \min\{z \in \Sigma^{\ell(n)} \mid z \notin \mathrm{KT}[n, 2^{s(i)\cdot n}]\} & \text{if } i = 1, \\ \min\{z \in \Sigma^{\ell(n)} \mid z \notin \mathrm{KT}[n, 2^{s(i)\cdot n} | z_1^n \cdots z_{i-1}^n]\} & \text{if } 1 < i \leq k(n). \end{cases}$$

Set $Z_i^n = z_1^n \cdots z_i^n \Sigma^{n - i\cdot\ell(n)}$ for all i with $1 \leq i \leq k(n)$.

Lemma 4.2.15 **[93]** *For every distribution μ in* **P**-comp, *there exist integers $c_0, n_0 > 0$ such that $\hat{\mu}(Z_i^n) \leq 2^{-(i-c_0)\ell(n)/2}$ for all $n \geq n_0$ and all i with $1 \leq i \leq k(n)$.*

Proof. The proof below is similar to that of Theorem 4.2.14. For a distribution $\mu \in$ **P**-comp, take T as in the proof of Theorem 4.2.14. This T also satisfies the inequality that $|\hat{\mu}(Z_i^n) - T(Z_i^n)| \leq 2^{-n}$ for all i with $1 \leq i \leq k(n)$. Assume that the value of $T(x\Sigma^m)$, where $|x| = n$, is computed in time $(n+m)^d + d$ for some positive integer d. We shall define c_0 and c' later. Choose n_0 large enough that the self-delimiting description of T is sufficiently small, and $n_0 > 4c'$, and $d + 2 \leq s(n_0)$.

The lemma is obviously true for each $n \geq n_0$ and each i with $1 \leq i \leq c_0$ since $\ell(n) > 0$ and

$$\hat{\mu}(Z_i^n) \leq \hat{\mu}(\Sigma^n) \leq 1 \leq 2^{-(i-c_0)\ell(n)/2}.$$

We shall show that the lemma also holds for all i with $k(n) \geq i > c_0$. Assume that the claim fails for some i and n. Now take the minimal integer i such that $\hat{\mu}(Z_i^n) > 2^{-(i-c_0)\ell(n)/2}$ for some $n \geq n_0$, and then take the minimal such n. Define a set A_i as

$$A_i = \{w \in \Sigma^{\ell(n)} \mid T(z_1^n \cdots z_{i-1}^n w \Sigma^{n-i\cdot\ell(n)}) + 2^{-n} > 2^{-\lceil (i-c_0)\ell(n)/2 \rceil}\}.$$

It is clear that $z_i^n \in A_i$ since $T(Z_i^n) + 2^{-n} > \hat{\mu}(Z_i^n) > 2^{-\lceil (i-c_0)\ell(n)/2 \rceil}$.

We claim that $\|A_i\| < 2^{(\ell(n)+4)/2}$. Assume otherwise. If $n \geq 2$, then

$$(i-1-c_0)\ell(n) + 5 \leq (k(n)-1)\ell(n) + 5 \leq n - 10\log n + 5 \leq n,$$

and hence, $2^{-(i-1-c_0)\ell(n)/2} \geq 2^{-(n-5)/2}$. Then, since $\ell(n) \leq n$,

$$\begin{aligned}
\hat{\mu}(Z_{i-1}^n) &\geq T(Z_{i-1}^n) - 2^{-n} > \|A_i\| \cdot (2^{-\lceil (i-c_0)\ell(n)/2 \rceil} - 2^{-n}) - 2^{-n} \\
&\geq 2^{(\ell(n)+2)/2}(2^{-((i-c_0)\ell(n)+2)/2} - 2^{-n}) - 2^{-n} \\
&= 2^{-(i-1-c_0)\ell(n)/2+1} - 2^{-n}2^{(\ell(n)+3)/2} \cdot 2 \\
&\geq 2^{-(i-1-c_0)\ell(n)/2+1} - 2^{-n+(n+3)/2+1} \\
&\geq 2 \cdot 2^{-(i-1-c_0)\ell(n)/2} - 2^{-(n-5)/2} \geq 2^{-(i-1-c_0)\ell(n)/2}.
\end{aligned}$$

This contradicts the minimality of i. Hence, we have $\|A_i\| < 2^{(\ell(n)+4)/2}$.

To reach a contradiction, it is suffices to show that $A_i \subseteq \mathrm{KT}[n, 2^{s(i)\cdot n} | z_1^n \cdots z_{i-1}^n]$. Now suppose $A_i = \{w_1, w_2, \ldots, w_m\}$, where $w_1 < w_2 < \cdots < w_m$, $m < 2^{(\ell(n)+4)/2}$. Consider the eth element w_e of A_i, where $1 \leq e < 2^{(\ell(n)+4)/2}$. Consider the following algorithm N which computes w_e:

begin deterministic algorithm for N
  input $\langle c, l, n, e, z\rangle$
  find i such that $|z| = (i-1)\cdot l$
  let $j = 0$
  for all string $y \in \Sigma^l$ **do**
    compute the value $T(zy\Sigma^{n-i\cdot l})$ (let this term to be t)
    if $t + 2^{-n} \leq 2^{-\lceil (i-c)l/2 \rceil}$ **then** go to $(*)$
    let $j := j + 1$
    if $j = e$ **then** output y
$(*)$ **end-for**
  output 0
end.

Let i_0 be an index of this machine N. The description of this algorithm needs only the index of T. Take c_0 large enough so that the length of this description i_0 satisfies $\log c_0 \geq 2\log i_0 + 12$. By our assumption, $c_0 < k(n)$.

The universal Turing machine U takes an input of the form $\langle i_0, c_0, \ell(n), n, e, z_1^n \cdots z_{i-1}^n \rangle$ and outputs w_e. The size of $\langle i_0, c_0, \ell(n), n, e \rangle$ is bounded by

$$\begin{aligned} & 2|i_0| + 2|c_0| + 2|\ell(n)| + 2|n| + |e| + 1 \\ & \quad \leq \quad 2\log i_0 + 2\log c_0 + 2\log \ell(n) + 2\log n + \frac{\ell(n)+4}{2} + 10 \\ & \quad \leq \quad 3\log c_0 + 2\log \ell(n) + 2\log n + \frac{\ell(n)}{2} \\ & \quad \leq \quad 3\log k(n)\ell(n) + 2\log n + \frac{\ell(n)}{2} \quad \leq \quad 5\log n + \frac{\ell(n)}{2} \\ & \quad \leq \quad \frac{\ell(n)}{2} + \frac{\ell(n)}{2} \quad \leq \quad \ell(n), \end{aligned}$$

since $c_0 \leq k(n)$ and $\ell(n) \geq 10 \log n$ for all $n \geq n_0$.

The running time of U on input $\langle i_0, c_0, \ell(n), n, e, z_1^n \cdots z_{i-1}^n \rangle$ is at most, for some constant $c' > 0$,

$$\begin{aligned} & c' \cdot 2^{\ell(n)} \left(\frac{\ell(n)+4}{2} + (n^d + d) + (n+2) + \frac{(i-c_0)\ell(n)}{2} + 1 \right) \\ & \qquad \leq \quad c' \cdot 2^{\ell(n)} \left(\frac{k(n)\ell(n)}{2} + 2n^d + (5+d) \right) \\ & \qquad \leq \quad 2^{\ell(n)} \cdot 4c' \cdot n^d \quad \leq \quad 2^n \cdot n^{d+1} \\ & \qquad \leq \quad 2^{\ell(n)} \cdot 2^{(d+1)\log n} \quad \leq \quad 2^{(d+2)\ell(n)} \\ & \qquad \leq \quad 2^{s(i)\cdot \ell(n)} \end{aligned}$$

since $i + 1 - c_0 \leq k(n)$, $4c' \leq n$, $\ell(n) \geq \log n$, and $4 + d \leq n^d$. Therefore, we conclude that $w_e \in \mathrm{KT}[n, 2^{s(i)\cdot n} | z_i^n \cdots z_{i-1}^n]$. □

4.2.3 Fault-Tolerance of Distributions

Suppose that we are going to solve a problem in reasonably short time with the help of communication with another supplementary source of information. Clearly our computation depends on the accuracy of incoming data from the source. Unless we can guarantee its accuracy (e.g., no cable breakdown or interference, etc.), it becomes important to make our computation robust and to make it tolerate any faults in data. The computation may of course require more time in case there is faulty information from the source.

In 1985, Schöning [90] introduced the notion of *robust machines* to model fault-tolerant computation and the notion of oracles *helping* such computation. We shall adapt his concept to our distribution setting and introduce a new concept, *fault-tolerance* of distributions, into average-case complexity theory.

For our purpose, we first introduce the concept of distributions *computable relative to* oracles.

Definition 4.2.16 (Relativized Computable Distributions) Let A be an oracle and let $\mathcal{C}$ be a complexity class.

1. A semi-distribution μ is *polynomial-time computable relative to* A ($\mathbf{P}^A$-computable, for short) if there exist a polynomial p and a deterministic oracle Turing machine M such that, on input $(x, 0^i)$, M^A

works in polynomial time and satisfies $|\mu(x) - M^A(x, 0^i)| \leq 2^{-i}$ for all strings $x \in \Sigma^*$ and all numbers $i \in \mathbb{N}$. Let $\mathbf{P}^A$-comp denote the set of all $\mathbf{P}^A$-computable distributions.

2. A semi-distribution μ is *polynomial-time computable relative to* $\mathcal{C}$ ($\mathbf{P}^{\mathcal{C}}$-computable, for short) if μ is $\mathbf{P}^A$-computable for some set A in $\mathcal{C}$. The notation $\mathbf{P}^{\mathcal{C}}$-comp denotes the union of all $\mathbf{P}^A$-comp for any A in $\mathcal{C}$.

We note, similar to the remark following Definition 4.2.2, that $\mathbf{P}^{\mathbf{E}}$-comp = $\mathbf{E}$-comp and $\mathbf{P}^{\mathbf{EXP}}$-comp = $\mathbf{EXP}$-comp.

Definition 4.2.17 (Fault Tolerance) Let A be a set and $\mathcal{C}$ be a complexity class. A distribution μ is in $\mathbf{P}^A_{help}$-comp if there exists a deterministic Turing machine M such that

(i) for every oracle O, $|\mu(x) - M^O(x, 0^i)| \leq 2^{-i}$ holds for all $i \in \mathbb{N}$; and

(ii) there exists a polynomial p satisfying $\mathrm{Time}^A_M(x, 0^i) \leq p(|x|, i)$ for all x and i.

The set A is said to *help* M. Let $\mathbf{P}^{\mathcal{C}}_{help}$-comp denote the collection of all distributions in $\mathbf{P}^A_{help}$-comp for some $A \in \mathcal{C}$.

It is obvious that $\mathbf{P}$-comp $\subseteq \mathbf{P}^A_{help}$-comp $\subseteq \mathbf{P}^A$-comp for every oracle A.

Lemma 4.2.18 $\mathbf{P}^{2^{\Sigma^*}}_{help}$-comp $\subseteq \mathbf{P}^{\mathbf{\Sigma}^{\mathrm{p}}_2}$-comp.

Proof. Assume that μ is in $\mathbf{P}^A_{help}$-comp for some A. There exist a deterministic oracle Turing machine M and a polynomial q which witness μ being in $\mathbf{P}^A_{help}$-comp. We shall define another machine M' that tries to exhaust all possible computation paths of M within time q. Here is the description of the machine.

begin nondeterministic algorithm for M'
  input $\langle x, 0^i \rangle$
  start the simulation of M on input x
  while the simulation **do**
    if M queries y **then** (nondeterministically) guess the oracle answer
    exit the loop **when** $q(|x|, i)$ steps are consumed
  end-while
  if M halts **then output** $M(x)$ **else output** λ
end.

Note that M' is polynomial-time bounded.

For each x, let $path_x$ be the minimal (code of) path p of the computation tree given by M' on input x such that p does not lead to the output λ. Let us consider the output of M' on input x along path $path_x$, say d_x. We get $|\mu(x) - d_x| \leq 2^{-i}$ because, otherwise, it violates the condition $|\mu(x) - M^C(x, 0^i)| \leq 2^{-i}$ if C is chosen so that $M^C(x, 0^i) = d_x$. Next define the desired set B as follows:

$$B = \{\langle x, 1^n\rangle \mid \text{the } i\text{th bit of the binary output } d_x \text{ of } M' \text{ on input } x \text{ on path } path_x \text{ is } 1\ \}.$$

The set B belongs to $\mathbf{\Sigma}_2^{\mathrm{p}}$ because we must choose the minimal paths first, and this needs two alternations of the existential state and the universal state. With the help of B as an oracle, we can compute the value $M^A(x, 0^i)$ in time polynomial in $|x|$ and i. Thus, μ is in $\mathbf{P}^{\mathbf{\Sigma}_2^{\mathrm{p}}}$-comp. □

It is unknown whether $\mathbf{P}_{help}^{2^{\Sigma^*}}$-comp $\subseteq \mathbf{P}^{\mathbf{NP}}$-comp. However, oracle sets in $\mathbf{UP} \cap$ co-$\mathbf{UP}$ do not increase the computational power of the robust machine.

Lemma 4.2.19 $\mathbf{P}_{help}^{\mathbf{UP}\cap\text{co-}\mathbf{UP}}$-comp $= \mathbf{P}^{\mathbf{UP}\cap\text{co-}\mathbf{UP}}$-comp.

Proof. It suffices to show that $\mathbf{P}^{\mathbf{UP}\cap\text{co-}\mathbf{UP}}$-comp $\subseteq \mathbf{P}_{help}^{\mathbf{UP}\cap\text{co-}\mathbf{UP}}$-comp. Let μ be any distribution in $\mathbf{P}^A$-comp for some $A \in \mathbf{UP} \cap$ co-$\mathbf{UP}$. There are two polynomial-time unambiguous Turing machines N_0 and N_1 computing A and $\overline{A}$, respectively. Let M be a deterministic polynomial-time oracle Turing machine computing μ relative to A.

We use a set called a *witness* of the accepting computation of N_0 and N_1. Set

$$Witness(A) = \{\langle x, s_i\rangle \mid \exists w \in \Sigma^{|x|}[\text{ and } N_0 \text{ accepts } x \text{ on path } w \text{ whose } i\text{th bit is } 1\ ]\}.$$

We wish to modify the machine M in order to compute the same distribution. Let $O = O_0 \oplus O_1$, and let us define the deterministic Turing machine that computes μ as follows:

begin deterministic algorithm for N with oracle O $(= O_0 \oplus O_1)$
  input x
  start the simulation of M on input x
  while the simulation **do**
    exit the loop **when** $p(|x|)$ steps are consumed
    halt the algorithm **when** an accepting or rejecting state is reached
    if y is queried **then** do the following:
      (i) query $\{\langle y, s_0\rangle, \langle y, s_1\rangle, \ldots, \langle y, s_{|x|}\rangle\}$ to oracles O_0 and O_1
      (ii) let $w_0 := O_0(\langle y, s_0\rangle) \cdots O_0(\langle y, s_{|x|}\rangle)$ and $w_1 := O_1(\langle y, s_0\rangle) \cdots O_1(\langle y, s_{|x|}\rangle)$
      (iii) **if** N_0 accepts y on path u_0 **then** let an oracle answer be "yes"
      (iv) **if** N_1 accepts y on path u_1 **then** let an oracle answer be "no"
        else go to $(*)$
  end-while
$(*)$ simulate M on input x using N_0 and N_1 as oracles
end.

It is clear that N computes the distribution μ, and if the set $Witness(A) \oplus Witness(\overline{A})$ is given as an oracle, then the running time of N with this oracle on input x is $O(\mathrm{Time}_M(x))$. □

4.3 Normalization of Semi-Distributions

In this section, we shall show how to normalize a semi-distribution to a full distribution. As the reader can see, there are several way to normalize semi-distributions. Here are two simple methods: for a given non-trivial semi-distribution μ, let

(i) $\hat{\mu}'(x) = \begin{cases} \hat{\mu}(x) & \text{if } x \neq \lambda, \\ 1 - \sum_{z:z\neq\lambda} \hat{\mu}(z) & \text{if } x = \lambda. \end{cases}$

(ii) $\hat{\mu}''(x) = \frac{1}{c} \cdot \hat{\mu}(x)$, where $c = \lim_{x\to\infty} \mu(x)$.

Both distributions μ' and μ'' need the computability of the limit $\lim_{x\to\infty} \mu(x)$. This is seen as follows. Assume that μ is **P**-computable (for simplicity, assume $\hat{\mu}(\lambda) = 0$). Assume that $\mu' \in$ **P**-comp. Let $c = \sum_{z:z\neq\lambda} \hat{\mu}(z)$. Since $c = \lim_{x\to\infty} \mu(x)$, c is **P**-computable. Thus, by Lemma 2.7.4, the inverse $\frac{1}{c}$ is also **P**-computable. Since $\mu''(x) = \frac{1}{c} \cdot \mu(x)$, μ'' becomes **P**-computable. Conversely, assume that $\mu'' \in$ **P**-comp. Then, c is **P**-computable since $c = \hat{\mu}(x)/\hat{\mu}''(x)$ for some x for which $\hat{\mu}''(x) > 0$. Note that $c = \lim_{x\to\infty} \mu(x) = \sum_{z:z\neq\lambda} \hat{\mu}(z)$. Thus, $\mu' \in$ **P**-comp.

One may raise the question: can all **P**-computable semi-distributions be normalized to some **P**-computable distributions by method (i) or (ii) ?. The answer is unfortunately negative.

Proposition 4.3.1 *There exists a* **P***-computable semi-distribution* μ *such that neither* μ' *nor* μ'' *is* **P***-computable.*

Proof. Take a tally set $A \subseteq \Sigma^+$ which is recursive but not **P**-printable. Since A is recursive, we choose a deterministic Turing machine M which, on input λ, produces a list of all strings of A (possibly with repetition).

Now we define the desired semi-distribution μ as

$$\hat{\mu}(x) = \begin{cases} 2^{-2i} & \text{if } x \in \{0\}^+ \text{ and } M \text{ produces } 0^i \text{ within } |x| \text{ steps}, \\ 0 & \text{otherwise.} \end{cases}$$

We first claim that μ is a **P**-computable semi-distribution.

$$\sum_x \hat{\mu}(x) = \sum_{k\in\mathbb{N}} \hat{\mu}(1^k) \leq \sum_{i=1}^{\infty} 2^{-2i} = \frac{1}{3}.$$

Set $c = \lim_{x\to\infty} \mu(x)$ and let $0.r$ be its binary representation. Notice that this representation is unique because, for every i, the $(2i+1)$th bit of r must be 0. Moreover,

$$A = \{0^i \mid \text{ the } 2i\text{-th bit of } r \text{ is } 1\ \}.$$

Assume that c is a **P**-computable real number. There exists a polynomial-time Turing machine N such that $|N(0^k) - c| \leq 2^{-k}$ for all $k \in \mathbb{N}$. Let $N'(0^k) = N(0^{2k+1})$. Machine N' is still polynomial-time bounded, and $N'(0^k)$ is an initial segment of r. Hence, for any sufficiently large n,

$$A \cap \Sigma^{\leq n} = \{0^i \mid \text{ the } 2i\text{th bit of } N'(0^{2n+2}) \text{ is } 1\ \}.$$

Thus, A is **P**-printable. This is a contradiction. □

We now know that we no longer guarantee the existence of normalized distributions. The reader may ask under what conditions can **P**-computable semi-distributions normalized. In the rest of this section, we shall discuss a sufficient condition for the normalization of computable semi-distributions.

First we consider a general case.

Lemma 4.3.2 *Let $\mu_1, \ldots, \mu_m$ be semi-distributions such that $\lim_{x\to\infty} \sum_{i=1}^{m} \mu_i(x) \leq 1$, and let $S_1, \ldots, S_m$ be m disjoint sets with $\Sigma^+ = \bigcup_{i=1}^{m} S_i$. Let h and p be increasing functions on $\mathbb{N}$, and p is time-constructible. Let*

$$\hat{\mu}(x) = \begin{cases} \hat{\mu}_i(x) & \text{if } x \in S_i \text{ for some } i \ (1 \leq i \leq m), \\ 1 - \sum_{z:z\neq\lambda} \hat{\mu}(z) & \text{if } x = \lambda. \end{cases}$$

Let $N_1, \ldots, N_m$ be $O(h(n))$-time bounded deterministic Turing machines. Assume that, for all $x \in \Sigma^*$ and $k \in \mathbb{N}$, the following two conditions (i) and (ii) hold:

(i) For all i with $1 \leq i \leq m$, $|N_i(x, 0^k) - \hat{\mu}_i(\{z \in S_i \mid z \leq x\})| \leq 2^{-k}$.

(ii) For all i with $1 \leq i \leq m$, $\hat{\mu}_i(S_i - \Sigma^{\leq p(k)}) \leq 2^{-k}$.

Then, μ is $O(h'(n))$-time computable, where $h'(n) = h(p(n + \mathrm{ilog}(m) + 1) + n + \mathrm{ilog}(m) + 1)$.

Proof. Take $\mu_1, \ldots, \mu_m$ and $S_1, \ldots, S_m$, and assume all the conditions of the lemma. For the sake of convenience, let $q(n) = n + \mathrm{ilog}(m) + 1$. Consider the following algorithm N.

begin deterministic Turing machine N
  input $(x, 0^k)$
  set $Result := 0$
  if $x = \lambda$ **then go to** $(*)$
  for $i = 1$ **to** m **do**
    simulate N_i on input $(x, 0^{q(k)})$
    set $Result := Result + N_i(x, 0^{q(k)})$
  end-for
  output $Result$ and **halt**
$(*)$ **for** $i = 1$ **to** m **do**
    simulate N_i on input $(1^{p(q(k))}, 0^{q(k)})$
    set $Result := Result + N_i(1^{p(q(k))}, 0^{q(k)})$
  end-for
  output $1 - Result$ and **halt**
end.

Take a constant $c > 0$ such that, for all appropriate i, $\mathrm{Time}_{N_i}(x, 0^k) \leq c \cdot h(|x| + k)$ and $\mathrm{Time}_{N'_i}(0^k) \leq c \cdot h(k)$. Then the running time of N on input $(x, 0^k)$ is calculated as follows: for some appropriate constants

c'' and d,

$$\begin{aligned}\mathrm{Time}_N(x,0^k) &\leq c''\left(\sum_{i=1}^m \mathrm{Time}_{N_i}(x,0^{q(k)}) + \sum_{i=1}^m \mathrm{Time}_{N_i}(1^{p(q(k))},0^{q(k)}) + 1\right)\\ &\leq c''\cdot m\cdot c\cdot (h(|x|+q(k)) + h(p(q(k))+q(k)))\\ &\leq d\cdot h(p(|x|+k+\mathrm{ilog}(m)+1)+n+\mathrm{ilog}(m)+1)\end{aligned}$$

since p and h are increasing. Hence, N is $O(h'(n))$-time bounded.

Next we show that N actually computes μ. Assume that $x \neq \lambda$.

$$\begin{aligned}\left|M(x,0^k) - \sum_{z:z\leq x}\hat{\mu}(z)\right| &= \left|\sum_{i=1}^m N_i(x,0^{q(k)}) - \sum_{i=1}^m\sum_{z:z\leq x}\hat{\mu}_i(z)\cdot[z\in S_i]\right|\\ &\leq \sum_{i=1}^m |N_i(x,0^{q(k)}) - \hat{\mu}(\{z\in S_i \mid z\leq x\})|\\ &\leq m\cdot 2^{-k-\mathrm{ilog}(m)-1} < 2^{-k}.\end{aligned}$$

For the case $x=\lambda$, we have

$$\begin{aligned}&|M(\lambda,0^k)-\hat{\mu}(\lambda)|\\ &= \left|\sum_{z:z\neq\lambda}\hat{\mu}(z) - \sum_{i=1}^m N_i(1^{p(q(k))},0^{q(k)})\right|\\ &\leq \left|\sum_{z:z\neq\lambda}\hat{\mu}(z) - \sum_{z:0<|z|\leq p(q(k))}\hat{\mu}(z)\right| + \left|\sum_{z:0<|z|\leq p(q(k))}\hat{\mu}(z) - \sum_{i=1}^m N_i(1^{p(q(k))},0^{q(k)})\right|\\ &\leq \sum_{i=1}^m\sum_{z:|z|>p(q(k))}\hat{\mu}_i(z)\cdot[z\in S_i] + \left|\sum_{i=1}^m\sum_{z:|z|\leq p(q(k))}\hat{\mu}_i(z)\cdot[z\in S_i] - \sum_{i=1}^m N_i(1^{p(q(k))},0^{q(k)})\right|\\ &< \sum_{i=1}^m \hat{\mu}_i(S_i - \Sigma^{\leq p(q(k))}) + \sum_{i=1}^m |\hat{\mu}_i(S_i\cap\Sigma^{\leq p(q(k))}) - N_i'(0^{p(q(k))})|\\ &\leq m\cdot 2^{-k-\mathrm{ilog}(m)-1} + m\cdot 2^{-k-\mathrm{ilog}(m)-1} < 2\cdot 2^{-k-1}\\ &= 2^{-k}.\end{aligned}$$

Therefore, μ is $O(h'(n))$-time computable. □

Consider, for example, the standard distribution ν_{stand}. When we set $p(n)=2^{n+1}-1$, the distribution ν_{stand} satisfies the convergence scheme $|\nu_{\mathrm{stand}}(1^{p(i)}) - \nu_{\mathrm{stand}}(1^{p(j)})| < 2^{-i}+2^{-j}$ for all $i,j\in\mathbb{N}$.

Corollary 4.3.3 *Let μ be a semi-distribution and let $p(n)$ be an increasing function on $\mathbb{N}$ bounded above by a polynomial in n. Let μ' and μ'' be the normalized distributions defined at the beginning of this section. If μ is* **P***-computable and $|\mu(1^{p(i)}) - \mu(1^{p(j)})| \leq 2^{-i}+2^{-j}$ for almost all $i,j\in\mathbb{N}$, then μ' and μ'' are both* **P***-computable.*

Proof. Assume that μ is a **P**-computable semi-distribution. Let M be a q-time bounded deterministic Turing machine computes μ, where q is an appropriate polynomial. To use Lemma 4.3.2, we set $S = \Sigma^*$ and let $N(x, 0^k) = M(x, 0^{k+1}) - M(\lambda, 0^{k+1})$ for all $k \in \mathbb{N}$. It sufficient to check conditions (i) and (ii) in the lemma. If both conditions are fulfilled, then Lemma 4.3.2 ensures the existence of the normalized distribution μ' that is $O(q(p(n+1)+n+1))$-time computable. Since $p(n)$ is bounded by some polynomial in n, clearly μ' is **P**-computable.

For (i), we have

$$\begin{aligned} |N(x,0^k) - \hat{\mu}(\{z \in S \mid z \leq x\})| &= |M(x,0^{k+1}) - M(\lambda,0^{k+1}) + \mu(x) - \mu(\lambda)| \\ &\leq |M(x,0^{k+1}) - \mu(x)| + |M(\lambda,0^{k+1}) - \mu(\lambda)| \\ &\leq 2^{-k-1} + 2^{-k-1} = 2^{-k}. \end{aligned}$$

For (ii), by our assumption, $|\mu(1^{p(i)}) - \mu(1^{p(j)})| \leq 2^{-i} + 2^{-j}$ for almost all $i, j \in \mathbb{N}$. By Lemma A.4, this is equivalent to the condition $|\lim_{x\to\infty} \mu(x) - \mu(1^{p(i)})| \leq 2^{-i}$. Hence, $\hat{\mu}(S - \Sigma^{\leq p(k)}) \leq 2^{-k}$. □

4.4 Samplable Distributions

Let us consider as a simple example the generation of an "occupied territory" on a finite square board (e.g., cf. [65]). First we randomly choose a nonnegative integer n, and define the "occupied territory" at stage 0 to be the center square of the $n \times n$ board. At stage i, a walker randomly chooses a starting point which is on the boundary of the board and walks to neighboring points at random. If the walker successfully reaches an adjacent point s_i of the occupied territory at stage $i-1$, then the territory is expanded to include the point s_i. We continue to the next stage. Allowing infinitely-many stages, we are able to consider the probability that a certain region becomes the occupied territory at some stage.

This type of (probability) distribution is called *samplable*, and the algorithm which produces instances under this distribution is called a *sampling algorithm* [9]. Instances of samplable distributions have often been observed in statistical physics. Samplable distributions are also of importance in cryptography. It is known that the existence of complex samplable distributions leads to the existence of pseudo-random generators (see [9, 40]).

4.4.1 Definition of Samplable Distributions

In 1990, Ben-David, Chor, Goldreich, and Luby [9] first formulated a notion of distributions which are sampled (or generated) by randomized algorithms in time polynomial in the length of their *output* on dyadic rational numbers. They coined the term, *polynomial samplable distributions* for such distributions.

In a recent work on pseudo-random number generators, Håstad *et al.* [40] also use an ensemble of "polynomial samplable" probability distributions. To cope with real-valued distributions, we use an approximation scheme and give a generalized definition of *t-time samplability*.

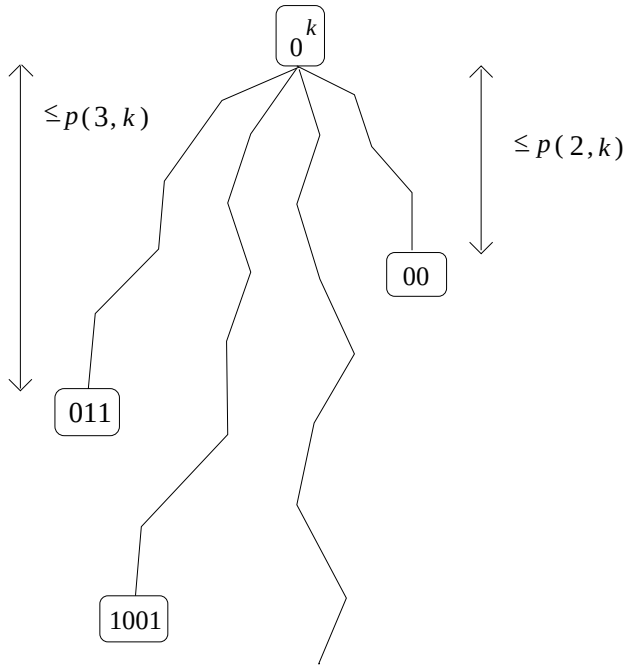

Figure 4.2: A computation tree of a sampling algorithm

Definition 4.4.1 (Samplable Distributions) Let t be a function on $\mathbb{R}^+$. A semi-distribution μ is *t-time samplable* if there exists a randomized Turing machine M (which does not necessarily halt on all computation paths), called a *sampling machine* or *generator*, such that

$$\left|\hat{\mu}(x) - \mathbf{Pr}_M[M \text{ on input } 0^i \text{ produces } x \text{ and halts within time } t(|x|,i)]\right| \leq 2^{-i} \qquad (*)$$

for all x and $i \in \mathbb{N}$. We say that M *samples* μ if M satisfies $(*)$. For a set $\mathcal{T}$ of functions, μ is *$\mathcal{T}$-time samplable* if μ is t-time samplable for some $t \in \mathcal{T}$.

Similarly, we can define μ to be *t-space samplable* (*$\mathcal{T}$-space samplable*, resp.) by requiring M to be t-space bounded ($\mathcal{T}$-space bounded, resp.).

An algorithm used for a sampling machine is called a *sampling algorithm*.

Definition 4.4.2 [9] A semi-distribution μ is *polynomial-time samplable* (**P**-samplable, for short) if there exists a polynomial q such that μ is q-time samplable. Denote by **P**-samp the set of all **P**-samplable distributions.

Figure 4.2 illustrates a computation tree of a sampling algorithm.

To distinguish our definition from Ben-David, Chor, Goldreich, and Luby's [9], we call their **P**-samplable distributions *strictly* **P**-*samplable*. We use the notation strict-**P**-samp to denote the collection of all strictly **P**-samplable distributions. Notice that all strictly **P**-samplable distributions are supportive.

Lemma 4.4.3 *For every* **P***-samplable distribution* μ*, there exists a distribution* ν *in* strict-**P**-samp *which* p*-dominates* μ*.*

Proof. Take an arbitrary **P**-samplable distribution μ. By definition, there exists a sampling machine M and a polynomial p such that, for all x and i,

$$\hat{\mu}(x) - \mathbf{Pr}_M[M(0^i) = x \text{ within time } p(|x|, i)\ ]| \leq 2^{-i}.$$

Replace i by $|x|$ in the above inequality, then we have

$$\hat{\mu} \leq 2^{-|x|} + \mathbf{Pr}_M[M(0^i) = x \text{ within time } p(|x|, |x|)\ ].$$

To simplify the description, write P_x for $\mathbf{Pr}_M[M(0^{|x|}) = x \text{ within time } p(|x|, |x|)\ ]$. We then have $\hat{\mu}(x) \leq 2^{-|x|} + P_x$.

Since the standard distribution is **P**-samplable, it has a its sampling machine, say M_0. We then define the desired semi-distribution ν to be sampled by the following sampling machine N:

begin sampling algorithm for N
  input λ (the empty string)
  choose a bit b at random
  if $b = 0$ **then** simulate M_0 on λ and **halt**
  generate a natural number n (actually generate s_n)
  simulate M on input 0^n
  if M enters a halting configuration **then** let x be its output and
    let t be the running time of M
  if $t \leq p(|x|, |x|)$ **then output** x **else output** λ
end.

Suppose that M outputs string x in time $p(|x|, |x|)$. In the case where $b = 1$ and $n = |x|$ are chosen, for some appropriate positive constant c, N can output x requiring its computation time to be at most $c \cdot (|x| + p(|x|, |x|) + 1)$, because N needs $O(|x|)$ steps to generate b and n and needs $O(p(|x|, |x|))$ steps for the simulation of N.

Now let $q(z) = c \cdot (z + p(z, z) + 1)$ and define ν as $\hat{\nu}(x) = \mathbf{Pr}_N[N(\lambda) = x \text{ within } q(|x|) \text{ steps}]$ for all x. Obviously this ν is a semi-distribution and also **P**-samplable.

We fix a nonempty string x arbitrarily. Note that the probability of generating n at random is exactly $2^{-2\mathrm{llog}(n)-1}$. Then the probability that $n = |x|$ holds and N outputs x within time $q(|x|)$, is at least

$$2^{-2\mathrm{llog}(n)-1} \cdot \mathbf{Pr}_M[M(0^{|x|}) = x \text{ within } p(|x|, |x|) \text{ steps}] = 2^{-2\mathrm{llog}(|x|)-1} \cdot P_x.$$

Hence, it follows that

$$\begin{aligned} \hat{\nu}(x) &= \mathbf{Pr}_N[N(\lambda) = x \text{ within } q(|x|) \text{ steps}] \\ &\geq \frac{1}{2} \cdot \hat{\nu}_{\mathrm{stand}}(x) + \frac{1}{2} \cdot 2^{-2\mathrm{llog}(|x|)-1} \cdot P_x \end{aligned}$$

$$\begin{aligned} &= 2^{-2\lfloor\log(|x|)\rfloor-2}\cdot 2^{-|x|} + 2^{-2\lfloor\log(|x|)\rfloor-2}\cdot P_x \\ &\geq \frac{1}{16(|x|+1)^2}\cdot(2^{-|x|}+P_x) \\ &\geq \frac{\hat{\mu}(x)}{16(|x|+1)^2}. \end{aligned}$$

Therefore, we conclude that $\mu \preceq^{\mathrm{p}} \nu$. □

Theorem 4.4.4 [9] *There exists an effective enumeration of all strictly* **P***-samplable semi-distributions. In particular, for each $k > 0$, there is an effective enumeration of all strictly $O(n^k)$-time samplable semi-distributions.*

Proof. First we effectively enumerate all randomized Turing machines (which may not halt on some computation paths). Let $\{M_i\}_{i\in\mathbb{N}}$ be such an enumeration. Also take an effective enumeration of all polynomials with positive integer coefficients, say $\{p_i\}_{i\in\mathbb{N}}$, such that each $p_i(z) \geq z$ for all z.

For each pair (i,j) of natural numbers, we shall consider the ith machine M_i and the jth polynomial p_j. We modify the machine as follows:

begin sampling algorithm for $M'_{\langle i,j\rangle}$
  input λ
  simulate M_i on input λ
  let x be the output of the machine M_i and
    let t be the running time of M_i
  if $t \leq p_j(|x|)$ **then output** x **else output** $x0^{t-|x|}$
end.

Note that a random seed generated by M'_i is exactly the same as that generated by M_i. Suppose that M_i outputs x. The running time of $M'_{\langle i,j\rangle}$ is $O(|x|+t+1)$. In the case where $t \leq p_j(|x|)$, the running time of $M'_{\langle i,j\rangle}$ is $O(p_j(|x|))$ because $p_j(n) \geq n$; otherwise, it is $O(t)$. Overall, the running time of $M'_{\langle i,j\rangle}$ is at most $c\cdot n$ steps in the length of its output.

It is easy to check that all **P**-samplable distributions appear in this enumeration. □

The reader will find an application of Theorem 4.4.4 in Section 5.3.

4.4.2 Invertibly Samplable Distributions

From a different point of view, Impagliazzo and Levin [44] defined "polynomial-time samplable" distributions to be of the form $\mu_{f^{-1}}$ for some $\mu \in$ **P**-comp and some $f \in$ **FP**. Following this definition, we can actually construct such a distribution that cannot belong to **P**-comp.

Although there is no proof separating the two notions of "polynomial-time samplability," in this thesis, we require f to be p-honest, and take the following weaker definition:

Definition 4.4.5 (Invertibly Samplable Distributions) [44]

1. A distribution μ is *invertibly polynomial-time samplable* (invertibly **P**-samplable, for short) if there exists a distribution $\nu \in$ **P**-comp and a p-honest function $f \in$ **FP** such that $\mu = \nu_{f^{-1}}$. Denote by **IP**-samp the set of all invertibly **P**-samplable distributions.

2. Let $\mathbf{IP}_1$-samp be the collection of all distributions of the form $\mu_{f^{-1}}$ for a distribution $\nu \in$ **P**-comp and a p-honest, one-one function f in **FP**.

Proposition 4.4.6 **[119]** **P**-comp $\subseteq \mathbf{IP}_1$-samp $\subseteq$ **IP**-samp.

Proof. For the first inclusion, use the identity function f. Obviously f is p-honest and one-one. Then, we have $\mu = \mu_{f^{-1}}$ for all distributions μ. The last inclusion is trivial. □

As shown in [111], in general, the feasible computability of $\mu_{f^{-1}}$ does not imply that of μ; namely, there are distributions μ which are not in **P**-comp, but $\mu_{f^{-1}}$ is in **P**-comp for the function $f(x) = 0^{|x|}$. Moreover, Wang and Belanger showed that, for every $\mu \in$ **P**-comp and every increasing, p-honest function $f \in$ **FP**, $\mu_{f^{-1}}$ belongs to **P**-comp [111].

We shall show the inclusion between **IP**-samp and **P**-samp.

Proposition 4.4.7 **[119]** **IP**-samp $\subseteq$ **P**-samp.

Proof. To establish the proposition, we shall generalize the proof of Theorem 7 in [9].

Assume that ν is in **IP**-samp. By definition, there exist a p-honest function $f \in$ **FP** and a deterministic polynomial-time Turing machine M such that $\nu = \mu_{f^{-1}}$ and $|\mu(x) - M(x, 0^i)| < 2^{-i}$ for all x and i. By Lemma 4.2.6, we can assume that $\lambda x.M(x, 0^k)$ is nondecreasing for each fixed k. We also assume that, for some polynomial p, $|x| \leq p(|f(x)|)$ and $|f(x)| \leq p(|x|)$ for all x.

For simplicity, write $\hat{M}(x, 0^k) = \sum_{z \in f^{-1}(x)} M'(z, 0^{p(|x|)+k-1})$, where $M'(x, 0^k) = M(x, 0^k) - M(x^-, 0^k)$. Note that $|\hat{\mu}(x) - M'(x, 0^k)| < 2^{-k+1}$. Since $|f(x)| \leq p(|x|)$, we have

$$|\hat{\nu}(x) - \hat{M}(x, 0^k)| \leq \sum_{z \in f^{-1}(x)} |\hat{\mu}(z) - M'(z, 0^{p(|x|)+k-1})| < 2^{p(|x|)} \cdot 2^{-p(|x|)-k} = 2^{-k}.$$

To complete the proof, we need to show that $\hat{M}(x, 0^k)$ can be computed by some sampling algorithm on input 0^k. Let us define the sampling algorithm N as follows:

begin sampling algorithm N
  input 0^k
  for $i = 1$ **to** ∞
    choose one bit b_i randomly
    let ρ_i be the real number identified with string $b_1 b_2 \cdots b_i$
    find the minimal string x by binary search such that

$$M(x^-, 0^{p(|x|)+k-1}) < \rho_i \leq M(x, 0^{p(|x|)+k-1})$$

if there is such an x **then output** $f(x)$ and **halt**

end-for

end.

It is not difficult to see that $\hat{M}(x, 0^k)$ is equal to the probability $\mathbf{Pr}[N(0^k) = x \text{ in time } q(|y|, k)]$ for some polynomial q. □

The converse of Proposition 4.4.7 is unlikely to hold; however, we can prove that every **P**-samplable distribution is p-dominated by some invertibly **P**-samplable distribution.

Lemma 4.4.8 **[119]** *For every $\mu \in$ **P**-samp, there exists a distribution $\nu \in$ **IP**-samp such that $\mu \preceq^{\mathrm{p}} \nu$.*

Proof. Let μ be a **P**-samplable distribution, and let M be a randomized Turing machine witnessing μ with a time-bound polynomial p. We modify the original machine M so that, at every configuration of M, there are exactly two nondeterministic choices. Hence, the length of a code which expresses a nondeterministic path of M on each input is at least the size of its output. Moreover, we assume without loss of generality that p is increasing.

We define a function f as

$$f(z) = \begin{cases} \text{output } x \text{ of } M \text{ on } 0^{2|x|} \text{ on path } z' \text{ in time } p(3|x|) & \text{if } z = z'1 \text{ and } x \text{ exists,} \\ \lambda & \text{if } z = z'1 \text{ but no such } x \text{ exists,} \\ z' & \text{if } z = z'0. \end{cases}$$

To see that f is **P**-computable, consider the following deterministic algorithm:

begin deterministic algorithm for f

input z

if $z = z'0$ **then output** z' **else** compute z' such that $z = z'1$

for $n = 1$ to $|z'|$

simulate M on input 0^{2n} on path z'

if M halts in time $p(3n)$ **then** let x be its output **else** go to $(*)$

if $|x| = n$ **then output** x and **halt**

$(*)$ **end-for**

output λ

end.

For each x, let A_x be the collection of all strings w such that, on input $0^{2|x|}$ on the computation path encoded by w, M halts in time $p(3|x|)$ and produces x. By our assumption, if $w \in A_x$, then $|w| \leq p(3|x|)$. Hence,

$$\mathbf{Pr}_M[M \text{ on } 0^{2|x|} \text{ produces } x \text{ in time } p(3|x|)] = \sum_w \frac{[w \in A_x]}{2^{|w|}}.$$

On the other hand, since M approximates $\hat{\mu}$, we then have $\hat{\mu}(x) \leq 2^{-2|x|} + \sum_w \frac{[w \in A_x]}{2^{|w|}}$.

Let $\hat{\nu}(x) = \hat{\nu}_{\mathrm{stand}}(\{w \mid f(w) = x\})$. Notice that $\hat{\nu}(\lambda) > 0$. Let c_0 be the minimal positive integer such that $c_0 \cdot \hat{\nu}(\lambda) \geq \hat{\mu}(\lambda)$. Then, it follows that

$$\begin{aligned}\hat{\nu}(x) &= \sum_w \hat{\nu}_{\mathrm{stand}}(w) \cdot [f(w1) = x] + \sum_w \hat{\nu}_{\mathrm{stand}}(w) \cdot [f(w0) = x] \\ &= \sum_w 2^{-|w|-2\mathrm{llog}(|w|)-1} \cdot [w \in A_x] + \hat{\nu}_{\mathrm{stand}}(x).\end{aligned}$$

Let $q(z) = 8(p(3z)+1)^2 + c_0$. In the following, we show that $q(|x|) \cdot \hat{\nu}(x) \geq \hat{\mu}(x)$. For $x = \lambda$, this is obviously the case. For the other strings $x \in \Sigma^+$,

$$\begin{aligned}q(|x|) \cdot \hat{\nu}(x) &= q(|x|) \cdot \sum_w 2^{-|w|-2\mathrm{llog}(|w|)-1} \cdot [w \in A_x] + q(|x|) \cdot \hat{\nu}_{\mathrm{stand}}(x) \\ &\geq \sum_{w:|w| \leq p(3|x|)} \frac{q(|x|)}{8(|w|+1)^2} \cdot \frac{[w \in A_x]}{2^{|w|}} + \frac{q(|x|)}{8(|x|+1)^2} \cdot \frac{1}{2^{|x|}} \\ &\geq \sum_w \frac{[w \in A_x]}{2^{|w|}} + \frac{1}{2^{|x|}} \;\geq\; \hat{\mu}(x).\end{aligned}$$

□

Unfortunately, we do not know whether **P**-samp $\subseteq^{\mathrm{p}}$ **IP**-samp, or whether we can replace **IP**-samp in Lemma 4.4.8 by $\mathbf{IP}_1$-samp.

Schuler and Watanabe [96] introduced an average version of **P**-samplable distributions. We give a brief definition here.

Definition 4.4.9 (Average Polynomial-Time Samplable Distributions) [96] A distribution μ is *average polynomial-time samplable* (average **P**-samplable, for short) if there exists a randomized Turing machine M and a polynomial p such that

(i) $|\hat{\mu}(x) - \mathbf{Pr}_M[M(0^i) = x] \leq 2^{-i}$ for any x and $i \in \mathbb{N}$; and

(ii) for every number $r > 0$ and every $n \in \mathbb{N}$,

$$\hat{\nu}_{\mathrm{tally}}(1^n) \cdot \mathbf{Pr}_s[M(0^i; s) \in \Sigma^n \wedge \mathrm{Time}_M(0^i; s) > p((n,i) \cdot r) \mid s \in \Gamma_M(0^i)] < 1/r.$$

Let **avP**-samp denote the set of all average **P**-samplable distributions.

We shall show below that average **P**-samplability is a natural extension of **P**-samplability.

Proposition 4.4.10 **P**-samp $\subseteq$ **avP**-samp.

Proof. Let μ be any **P**-samplable distribution. Consider a polynomial p and a randomized Turing machine M which, on input 0^i, samples $\mu(x)$ in $p(|x|, i)$ steps; that is,

$$|\hat{\mu}(x) - \mathbf{Pr}_M[M(0^i) = x \text{ in time } p(|x|, i)\ ]| \leq 2^{-i}$$

for all $i \in \mathbb{N}$ and $x \in \Sigma^*$.

To show the average **P**-samplability of μ, we must define another sampling machine that samples μ in polynomial time on μ-average with respect to the length of its output. Such a sampling machine is defined as follows:

begin sampling algorithm for N
  input 0^i
  simulate M on input 0^i
  if M reaches a halting state **then** do the following:
    let x be its output
    let s be the random seed generated so far
    if $\mathrm{Time}_M(0^i; s) \leq p(|x|, i)$ **then output** x and **halt**
    for $i = 1$ **to** ∞ **do**
    flip a fair coin
    end-for
end.

By the simulation, it follows that

$$\mathbf{Pr}_M[M(0^i) = x \text{ in time } p(|x|, i)\ ] = \mathbf{Pr}_s[N(0^i; s) = x \mid s \in \Gamma_N(0^i)].$$

The running time of N on input 0^i along with random seed s, is the same as that of M on 0^i with s with a constant factor.Thus, for an appropriate constant $c > 0$, it holds that

$$\mathbf{Pr}_s[N(0^i : s) \in \Sigma^n \wedge \mathrm{Time}_N(0^i; s) > c \cdot p(n, i) \mid s \in \Gamma_M(0^i)] = 0,$$

which implies that μ is average **P**-samplable. □

4.4.3 Closure Properties of Samplable Distributions

This subsection will discuss several properties of **P**-samplable and invertibly **P**-samplable distributions.

Lemma 4.4.11 *For every distribution μ and every function f in* **FP**, *if $\mu \in$* **IP**-samp *and f is p-honest, then $\mu_{f^{-1}}$ also belongs to* **IP**-samp.

Proof. Let f be p-honest and in **FP**. Assume that $\mu = \nu_{g^{-1}}$ for some $\nu \in$ **P**-comp and p-honest $g \in$ **FP**. Then,

$$\begin{aligned} \hat{\mu}_{f^{-1}}(x) &= \hat{\mu}(\{z \mid f(z) = x\}) = \hat{\nu}(\{w \mid \exists z[g(w) = z \wedge f(z) = x]\}) \\ &= \hat{\nu}(\{w \mid f \circ g(w) = x\}) = \hat{\nu}_{(f \circ g)^{-1}}(x). \end{aligned}$$

Since $f \circ g$ is p-honest and in **FP**, $\mu_{f^{-1}}$ belongs to **IP**-samp. □

Corollary 4.4.12 *For every $\mu \in$ **P**-samp and every p-honest function $f \in$ **FP**, there exists a $\nu \in$ **P**-samp such that $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$.*

Proof. Assume that $\mu \in$ **P**-samp. By Lemma 4.4.8, we can take a distribution μ' from **IP**-samp such that $\mu \preceq^{\mathrm{p}} \mu'$. By Lemma 3.4.10, $\mu \preceq^{\mathrm{p}} \mu'$ implies $\mu_{f^{-1}} \preceq^{\mathrm{p}} \mu'_{f^{-1}}$. Now let $\nu = \mu'_{f^{-1}}$. By Lemma 4.4.11, we conclude that ν belongs to **IP**-samp. □

This corollary will be used to prove Proposition 7.5.10.

The following theorem is similar to the fact that $\mathbf{BPP}^{\mathbf{BPP}} = \mathbf{BPP}$ (see, e.g., [123]). It states that $\mathbf{P}^{\mathbf{BPP}}$-samp is p-equal to **P**-samp. However, the proof requires complex analysis of a randomized algorithm.

Theorem 4.4.13 $\mathbf{P}^{\mathbf{BPP}}$-samp $\cong^{\mathrm{p}}$ **P**-samp.

Proof. Since clearly **P**-samp $\subseteq \mathbf{P}^{\mathbf{BPP}}$-samp, it suffices to show that $\mathbf{P}^{\mathbf{BPP}}$-samp $\subseteq^{\mathrm{p}}$ **P**-samp. Consider a distribution μ and assume that μ is in $\mathbf{P}^{\mathbf{BPP}}$-samp. There exists a Turing machine M and an oracle $A \in \mathbf{BPP}$ such that $|\hat{\mu}(x) - \mathbf{Pr}_M[M^A(0^k) = x \text{ in time } q(|x|,k)]| \leq 2^{-k}$ for all $k \in \mathbb{N}$, where q is an appropriate polynomial. We define another set A' as follows. Let $A' = \{x10^m \mid x \in A, m \in \mathbb{N}\}$. It is easy to see that $A' \in \mathbf{BPP}$. Let N be a polynomial-time probabilistic Turing machine such that

$$\mathbf{Pr}_N[N(x10^m) = A'(x10^m)] \geq 1 - 2^{-|x|-m-1}$$

(by the Amplification Lemma). Let $p(n) = 2n + 4$ for all $n \in \mathbb{N}$.

Let us consider the following randomized Turing machine M':

begin randomized algorithm for M'
  input 0^i
  generate a natural number n_0 at random
  let $count := 0$
  for $n = n_0$ **to** ∞ **do**
    while the simulation of M on input $0^{i+p(n)}$
      if M queries y **then** simulate N on input $y10^{q(n,i+p(n))-|y|+p(n)}$
        and let $count := count + 1$
    end-while
    (assume that M halts and writes down a string x)
    if $count < q(n, i + p(n))$ **then** flip a fair coin m times,
      where $m = (q(n, i + p(n)) + i + p(n))(q(n, i + p(n)) - count)$
    if $|x| = n$ **then output** x and **halt**
(∗) **end-for**
end.

Let $\overline{\rho}_x{}^i$ be the overall probability that x is generated by the algorithm when input 0^i is given. For simplicity, write $\sigma_x^i = \mathbf{Pr}_M[M^A(0^i) = x \text{ in time } q(|x|, i)]$. Remember that $\hat{\mu}(x) = \lim_{i\to\infty} \sigma_x^i$ holds for all x. It follows that $|\sigma_x^i - \sigma_x^j| \leq 2^{-i} + 2^{-j}$.

Our goal is to show that, for any string x,

(i) $\frac{1-2^{-i-p(n)}}{8(n+1)^2} \cdot \sigma_x^{i+p(n)} \leq \overline{\rho}_x{}^i \leq 2 \cdot \sigma_x^{i+p(n)}$ for almost all $i \in \mathbb{N}$; and

(ii) $|\overline{\rho}_x{}^i - \overline{\rho}_x{}^j| \leq 2^{-i} + 2^{-j}$ for almost all $i, j \in \mathbb{N}$.

By Lemma A.4, (ii) implies the existence of the limit $\lim_{i\to\infty} \overline{\rho}_x{}^i$. We then set $\hat{\nu}(x) = \lim_{i\to\infty} \overline{\rho}_x{}^i$. The condition (i) implies that

$$\frac{\hat{\mu}(x)}{8(n+1)^2} = \lim_{i\to\infty} \frac{1-2^{-i-p(n)}}{8(n+1)^2} \cdot \sigma_x^{i+p(n)} \leq \hat{\nu}(x) \leq 2 \cdot \lim_{i\to\infty} \sigma_x^{i+p(n)} = 2 \cdot \hat{\mu}(x).$$

Therefore, μ is p-equivalent to ν.

Fix x and let $n = |x|$. Let ρ_x^i denote the probability that the algorithm outputs x after integer n is already generated. This probability ρ_x^i is the product of the probabilities that N correctly computes all the query strings made by M, and thus is at least

$$\prod_{j=1}^{q(n,i+p(n))} \left(1 - 2^{-q(n,i+p(n))-i-p(n)}\right) \geq \left(1 - 2^{-q(n,i+p(n))-i-p(n)}\right)^{q(n,i+p(n))} \geq 1 - 2^{-i-p(n)-1},$$

where the last inequality holds by Lemma A.5. Then, we have

$$\sigma_x^{i+p(n)} \cdot (1 - 2^{-i-p(n)}) \leq \rho_x^i \leq \sigma_x^{i+p(n)}.$$

Now we show that $|\rho_x^i - \rho_x^j| \leq 2^{-p(n)+1}(2^{-i} + 2^{-j})$. Note that $\sigma_x^{i+p(n)} \leq 1$.

$$\begin{aligned}
|\rho_x^i - \rho_x^j| &\leq \max\{|\sigma_x^{i+p(n)} - \sigma_x^{j+p(n)} \cdot (1 - 2^{-i-p(n)})|, |\sigma_x^{j+p(n)} - \sigma_x^{i+p(n)} \cdot (1 - 2^{-j-p(n)})|\} \\
&\leq |\sigma_x^{i+p(n)} - \sigma_x^{j+p(n)}| + \max\{\sigma_x^{i+p(n)} \cdot 2^{-i-p(n)}, \sigma_x^{j+p(n)} \cdot 2^{-j-p(n)}\} \\
&\leq 2^{-i-p(n)} + 2^{-j-p(n)} + \max\{2^{-i-p(n)}, 2^{-j-p(n)}\} \\
&\leq 2^{-p(n)+1} \cdot (2^{-i} + 2^{-j}).
\end{aligned}$$

Let $\tau_n^i = \sum_{x:|x|=n} \rho_x^i$. The probability $\bar{\rho}_x^i$ is equivalent to the probability that, for every $k \leq n$, k is first generated, and the algorithm fails to halt until it reaches stage n of the **for**-loop and finally writes x. For each k, $0 \leq k \leq n$, with probability $2^{-2\mathrm{llog}(k)-1}$, the algorithm generates k; it eventually reaches n and with probability ρ_x^i, it outputs x. Hence, if $0 \leq k < n$, then the probability that x is an output is $2^{-2\mathrm{llog}(k)-1} \cdot \rho_x^i \cdot \prod_{j=k}^{n-1}(1 - \tau_j^i)$, and if $k = n$, then this probability is exactly $2^{-2\mathrm{llog}(n)-1} \cdot \rho_x^i$. Overall, we have

$$\overline{\rho}_x{}^i = \sum_{k=0}^{n-1} \left(\frac{\rho_x^i}{2^{2\mathrm{llog}(k)+1}} \prod_{j=k}^{n-1} (1 - \tau_j^i) \right) + \frac{\rho_x^i}{2^{2\mathrm{llog}(n)+1}}.$$

Using the fact that $\prod_{j=k}^{n-1}(1 - \tau_j^i) \leq 1$, we have

$$\frac{1-2^{-i-p(n)}}{8(n+1)^2} \cdot \sigma_x^{i+p(n)} \leq \frac{\rho_x^i}{2^{2\mathrm{llog}(n)+1}} \leq \overline{\rho}_x{}^i \leq \rho_x^i \cdot \sum_{k=0}^{n} \frac{1}{2^{2\mathrm{llog}(k)+1}} < \frac{\pi^2}{6} \rho_x^i < 2 \cdot \sigma_x^{i+p(n)}.$$

The absolute value of the difference between τ_x^i and τ_x^j is now easy to calculate, as shown below:

$$\begin{aligned} |\tau_n^i - \tau_n^j| &= \left| \sum_{x:|x|=n} \rho_x^i - \sum_{x:|x|=n} \rho_x^j \right| \leq \sum_{x:|x|=n} |\rho_x^i - \rho_x^j| \\ &\leq 2^n \cdot 2^{-p(n)+1} \cdot (2^{-i} + 2^{-j}) = 2^{-p(n)+n+1} \cdot (2^{-i} + 2^{-j}). \end{aligned}$$

Finally we shall show that $|\overline{\rho}_x{}^i - \overline{\rho}_x{}^j| \leq 2^{-i} + 2^{-j}$. We start with the following fragment. Note that $\left|\prod_{l=k}^{n-1}(1-\tau_l^i) - \prod_{l=k}^{n-1}(1-\tau_l^j)\right| \leq \sum_{l=k}^{n-1} |\tau_l^i - \tau_l^j|$ by Lemma A.5. Using $\rho_x^i \leq 1$ and $\prod_{j=k}^{n-1}(1-\tau_j^i) \leq 1$, we have

$$\begin{aligned} \left|\rho_x^i \prod_{l=k}^{n-1}(1-\tau_l^i) - \rho_x^j \prod_{l=k}^{n-1}(1-\tau_n^j)\right| &\leq \rho_x^i \cdot \left|\prod_{l=k}^{n-1}(1-\tau_l^i) - \prod_{l=k}^{n-1}(1-\tau_l^j)\right| + |\rho_x^i - \rho_x^j| \cdot \prod_{l=k}^{n-1}(1-\tau_l^j) \\ &\leq \sum_{l=k}^{n-1} |\tau_l^i - \tau_l^j| + |\rho_x^i - \rho_x^j| \\ &\leq 2^{-p(n)+n+1} \cdot (2^{-i} + 2^{-j}) + 2^{-p(n)+1} \cdot (2^{-i} + 2^{-j}) \\ &\leq (n-k) \cdot 2^{-p(n)+1} \cdot (2^n + 1) \cdot (2^{-i} + 2^{-j}). \end{aligned}$$

Using this inequality, we obtain

$$\begin{aligned} |\overline{\rho}_x{}^i - \overline{\rho}_x{}^j| &\leq \sum_{k=0}^{n-1} \frac{1}{2^{2\lfloor \log(k)+1}} \left|\rho_x^i \prod_{l=k}^{n-1}(1-\tau_l^i) - \rho_x^j \prod_{l=k}^{n-1}(1-\tau_n^j)\right| + \frac{|\rho_x^i - \rho_x^j|}{2^{2\lfloor \log(n)+1}} \\ &\leq \sum_{k=0}^{n-1} \frac{n-k}{2(k+1)^2} \left(2^{-p(n)+1} \cdot (2^n+1) \cdot (2^{-i} + 2^{-j})\right) + \frac{1}{2(n+1)^2} \cdot 2^{-p(n)+1} \cdot (2^{-i} + 2^{-j}) \\ &\leq (2^{-i} + 2^{-j}) \cdot 2^{-p(n)+1} \cdot \left[\sum_{k=0}^{n-1} \frac{n-k}{2(k+1)^2} \cdot (2^n+1) + \frac{1}{2(n+1)^2}\right] \\ &\leq (2^{-i} + 2^{-j}) \cdot 2^{-p(n)+1} \left[n(n-1)(2^n+1) + \frac{1}{(n+1)^2}\right] \\ &\leq (2^{-i} + 2^{-j}) \cdot 2^{-p(n)+1} \cdot (n^2 \cdot 2^{n+1} + 2) \leq (2^{-i} + 2^{-j}) \cdot 2^{-p(n)+1} \cdot 2n^2 \cdot 2^{n+1} \\ &\leq (2^{-i} + 2^{-j}) \cdot 2^{2n+3} \cdot 2^{-p(n)+1} \leq 2^{-i} + 2^{-j} \end{aligned}$$

since $2 \cdot n^2 \leq 2^{n+2}$ for all $n \in \mathbb{N}$. □

4.5 The P-comp = P-samp Question

We have seen two categories of feasible distributions, **P**-computable and **P**-samplable distributions, both of which have very different characteristics. It is natural to raise the question of whether these notions are truly different. The first answer was given by Ben-David, Chor, Goldreich, and Luby [9] who showed that **P**-samp $\neq$ **P**-comp unless **NP** collapses to **P**. Later Milterson [74] pointed out that **P** = **PP** is a sufficient and necessary condition for **P**-comp = **P**-samp, and its proof appeared in Yamakami [119]. This section shows that **P**-samplable distributions are **P**-computable if and only if **P** = **PP**. Based on the common belief that **P** $\neq$ **PP**, it seems unlikely that **P**-comp equals **P**-samp.

We shall introduce another category of distributions, the so-called #**P**-*computable distributions* introduced by Schuler and Watanabe [96], which seem to have more computational power than **P**-samplable distributions. Again we modify their definition to fit our approximation scheme.

Definition 4.5.1 (#P-Computable Distributions) cf. [96] A distribution μ is #**P**-*computable* if there exist a function $f \in \#\mathbf{P}$ and a polynomial p such that $\left|\hat{\mu}(x) - \frac{f(x,0^i)}{2^{p(|x|,i)}}\right| \leq 2^{-i}$ for all $x \in \Sigma^*$ and $i \in \mathbb{N}$. Denote by #**P**-comp the set of all #**P**-computable distributions.

We first show that the density function $\hat{\mu}$ in the above definition can be replaced by its distribution μ.

Lemma 4.5.2 *For a distribution $\mu \in \#\mathbf{P}$-comp, there exist a function $f \in \#\mathbf{P}$ and a polynomial q such that $\left|\mu(x) - \frac{f(x,0^i)}{2^{q(|x|,i)}}\right| \leq 2^{-i}$ for all x and i.*

Proof. Assume that μ is #**P**-computable. By the definition of #**P**-computability, there are a set $A \in \mathbf{P}$ and two polynomials p and q such that $\left|\hat{\mu}(x) - \frac{f(x,0^i)}{2^{q(|x|,i)}}\right| \leq 2^{-i}$ and $f(x,0^i) = \|\{y \mid |y| = p(|x|,i) \wedge \langle x, 0^i, y\rangle \in A\}\|$. Without loss of generality, we assume that p and q are increasing.

We first show that the function g defined as $g(x,0^i) = 2^{q(|x|,i)} \cdot \sum_{z \leq x} \frac{f(z,0^i)}{2^{q(|z|,i)}}$ is in #**P**. To see this, we define another set A' as follows:

$$A' = \{\langle x, z10^{n-|z|}y\rangle \mid |x| = n, z \leq x, \exists\{y_1, \ldots, y_{q(n,i)}\}[y = y_1 \cdots y_{q(n,i)}$$
$$\wedge \forall j[1 \leq j \leq q(n,i) \rightarrow \exists \tilde{y}_j[y_j = \tilde{y}_j 10^{p(n,i)-|\tilde{y}_j|} \wedge |\tilde{y}_j| = p(|z|,i) \wedge \langle x, 0^i, \tilde{y}_j\rangle \in A]]]\}.$$

Let $p'(n,i) = n + 1 + (p(n,i)+1)q(n,i)$. It is not difficult to prove that g can be characterized as

$$g(x,0^i) = \|\{u \mid |u| = p'(|x|,i) \wedge \langle x, u\rangle \in A'\}\|,$$

and thus g is in #**P**.

Now let $g'(x,0^i) = g(x,0^{|x|+i+1})$ and $q'(n,i) = q(n,n+i+1)$. Then, for any string x of length n, we have:

$$\begin{aligned}\left|\mu(x) - \frac{g'(x,0^i)}{2^{q'(n,i)}}\right| &= \left|\sum_{z \leq x} \hat{\mu}(z) - \sum_{z \leq x} \frac{f(x,0^{n+i+1})}{2^{q'(|z|,n+i+1)}}\right| \\ &\leq \sum_{z \leq x} \left|\hat{\mu}(z) - \frac{f(z,0^{n+i+1})}{2^{q(|z|,n+i+1)}}\right| \\ &\leq 2^{n+1} \cdot 2^{-(n+i+1)} = 2^{-i}.\end{aligned}$$

□

Proposition 4.5.3 $\#\mathbf{P}\text{-comp} \subseteq \mathbf{P}^{\#\mathbf{P}}\text{-comp} = \mathbf{P}^{\mathbf{PP}}\text{-comp}$.

Proof. Let μ be a distribution in #**P**-comp. Using Lemma 4.5.2, take a function $f \in \#\mathbf{P}$ and a polynomial q such that $\left|\mu(x) - \frac{f(x,0^i)}{2^{q(|x|,i)}}\right| \leq 2^{-i}$. Since f is computable in polynomial time relative to f itself,

μ is $\mathbf{P}$-computable relative to f. Hence, $\#\mathbf{P}\text{-comp} \subseteq \mathbf{P}^{\#\mathbf{P}}\text{-comp}$. Since $\mathbf{FP}^{\#\mathbf{P}} = \mathbf{FP}^{\mathbf{PP}}$ by Lemma 2.5.3, it follows that $\mathbf{P}^{\#\mathbf{P}}\text{-comp} = \mathbf{P}^{\mathbf{PP}}\text{-comp}$. □

The main theorem of this section is:

Theorem 4.5.4 **[74, 119]** *The following five statements are equivalent:*

1. $\mathbf{P} = \mathbf{PP}$.
2. $\mathbf{P}\text{-comp} = \#\mathbf{P}\text{-comp}$.
3. $\mathbf{P}\text{-comp} = \mathbf{P}\text{-samp}$.
4. $\mathbf{P}\text{-comp} = \mathbf{IP}\text{-samp}$.
5. $\mathbf{P}\text{-comp} = \mathbf{IP}_1\text{-samp}$.

The theorem immediately follows from the proposition and two lemmas below.

Proposition 4.5.5 **[119]** $\mathbf{P}\text{-samp} \subseteq \#\mathbf{P}\text{-comp}$.

Proof. Assume that μ is $\mathbf{P}$-samplable and is witnessed by a sampling algorithm M and a polynomial p. Without loss of generality, we assume that every path of M on 0^i which outputs x halts in exactly $p(|x|, i)$ steps. Let $f(x, 0^i)$ be the number of computation paths y such that M on 0^i outputs x and halts on path y in time $p(|x|, i)$. Clearly $f \in \#\mathbf{P}$ since each path of M on 0^i is bounded by $p(|x|, i)$. It is easy to see that the probability that $M(0^i)$ outputs x and halts in time $p(|x|, i)$ equals $f(x, 0^i)/2^{p(|x|,i)}$. Hence, μ turns out to be $\#\mathbf{P}$-computable. □

The converse inclusion, $\#\mathbf{P}\text{-comp} \subseteq \mathbf{P}\text{-samp}$, is an open question. The best known result is due to Schuler and Watanabe [96] that every $\#\mathbf{P}$-computable conditional distribution can be approximated within a polynomial factor by some sampling algorithm in time polynomial in the length of outputs with nonadaptive queries to an $\mathbf{NP}$ oracle. This will be shown as Proposition 4.7.6.

The next lemma establishes a basic relationship between $\#\mathbf{P}$ and $\#\mathbf{P}\text{-comp}$.

Lemma 4.5.6 $\mathbf{P} = \mathbf{PP}$ *implies* $\mathbf{P}\text{-comp} = \#\mathbf{P}\text{-comp}$.

Proof. This lemma is an immediate consequence of Proposition 4.5.3. However, we here show this lemma in a more direct way.

Let us assume that $\mathbf{P} = \mathbf{PP}$. This is equivalent to the assumption $\mathbf{FP} = \#\mathbf{P}$ by Lemma 2.5.3. For an arbitrary distribution μ in $\#\mathbf{P}\text{-comp}$, assume that there exists a function $f \in \#\mathbf{P}$ and a nondecreasing polynomial p such that $\left|\hat{\mu}(x) - \frac{f(x,0^i)}{2^{p(|x|,i)}}\right| < 2^{-i}$ for all x and $i \in \mathbb{N}$. Now we show that μ is computable by some deterministic Turing machine in polynomial time.

Define $g(x,0^i) = \sum_{z \le x} h(z,x,0^i)$, where $h(z,x,0^i) = f(z,0^{|x|+i}) \cdot 2^{p(|x|,|x|+i)-p(|z|,|x|+i)}$. Since $g \in \#\mathbf{P}$, it follows from our assumption that $g \in \mathbf{FP}$. We then define the deterministic Turing machine M such that $M(x,0^i)$ outputs $g(x,0^i)/2^{q(|x|,i)}$, where $q(n,i) = p(n,n+i)$. Thus, M satisfies:

$$\begin{aligned} |\mu(x) - M(x,0^i)| &= \left|\mu(x) - \frac{g(x,0^i)}{2^{q(|x|,i)}}\right| = \left|\mu(x) - \sum_{z \le x} \frac{h(z,x,0^i)}{2^{q(|x|,i)}}\right| \\ &\le \sum_{z \le x} \left|\hat{\mu}(z) - \frac{f(z,0^{|x|+i})}{2^{p(|z|,|x|+i)}}\right| \le 2^{|x|} \cdot 2^{-|x|-i} \\ &= 2^{-i}. \end{aligned}$$

Hence, $\mu \in \mathbf{P}$-comp. This completes the proof. □

In the following lemma, we prove that $\mathbf{IP}_1$-samp $= \mathbf{P}$-comp implies $\mathbf{P} = \mathbf{PP}$.

Lemma 4.5.7 **[119]** *If* $\mathbf{IP}_1$-samp $= \mathbf{P}$-comp, *then* $\mathbf{P} = \mathbf{PP}$.

Proof. Let us assume that $\mathbf{IP}_1$-samp $= \mathbf{P}$-comp: namely, for any $\mu \in \mathbf{P}$-comp and any one-one, p-honest, $\mathbf{P}$-computable function f, the distribution $\mu_{f^{-1}}$ is $\mathbf{P}$-computable. We shall show that $\mathbf{FP} = \#\mathbf{P}$, which is equivalent to $\mathbf{P} = \mathbf{PP}$.

Given a set A in $\mathbf{P}$ and a polynomial p, we set $g(x) = \|\{y \in \Sigma^{p(|x|)} \mid xy \in A\}\|$. We can assume without loss of generality that p is strictly increasing. We want to show that $g \in \mathbf{FP}$.

Now take the standard distribution ν_{stand} and define the one-one, $\mathbf{P}$-computable function f as follows:

$$f(xy) = \begin{cases} 0xy & \text{if } xy \in A \text{ and } |y| = p(|x|), \\ 1xy & \text{if } xy \notin A \text{ and } |y| = p(|x|), \\ xy & \text{otherwise.} \end{cases}$$

We also define the invertibly $\mathbf{P}$-samplable distribution η by $\hat{\eta} = \lambda x.\hat{\nu}_{\mathrm{stand}}(f^{-1}(x))$. By our assumption, η is $\mathbf{P}$-computable. For the function g, we have the following simple equation:

$$2^{-r(|x|)-2\mathrm{llog}(r(|x|))-1} \cdot g(x) = \sum_{y:|y|=p(|x|)} \hat{\nu}_{\mathrm{stand}}(f^{-1}(0xy)) = \eta(0x1^{p(|x|)}) - \eta(0x^-1^{p(|x|)}),$$

where $r(n) = n + p(n) + 1$. Therefore, g is $\mathbf{P}$-computable. □

We combine the above lemmas and propositions to complete the main theorem.

4.6 Universal Distributions

A distribution μ is called *universal* if, for every recursive distribution ν, there exists a constant $c > 0$ such that $c \cdot \hat{\mu}(x) \ge \hat{\nu}(x)$ holds for all strings x (see e.g., [61, 74]). These universal distributions are known to be *malign*: that is, average-case complexity equals worst-case complexity [61]. It is also known that there is no recursive universal distribution (see e.g., [61]).

This section will introduce a slightly weaker notion of universal distributions, called *p-universal distributions*, and shows that there is no p-universal distribution in **P**-comp, which is due to Schuler [95]. This result is difficult to extend to the even weaker notion of $O(f)$-universal distributions.

We begin with the formal definition.

Definition 4.6.1 ($\mathcal{T}$-universal distributions) Let $\mathcal{F}$ be a set of distributions and $\mathcal{T}$ a set of functions from Σ^* to $\mathbb{R}^+$. A distribution μ is called *$\mathcal{T}$-universal for $\mathcal{F}$* if

(i) $\mu \in \mathcal{F}$; and

(ii) for every $\nu \in \mathcal{F}$, there exists a function $t \in \mathcal{T}$ such that $t(x) \cdot \hat{\mu}(x) \geq \hat{\nu}(x)$ for all strings x.

In particular, if $\mathcal{T}$ is the set of p-bounded functions, then μ is called *p-universal*.

The following theorem of Schuler [95] is a negative reply to the question of whether p-universal **P**-computable distributions exist for **P**-comp.

Theorem 4.6.2 **[95]** *There is no single* **P***-computable distribution which avp-dominates all* **P***-computable distributions. Hence, no p-universal distributions exist in* **P**-comp.

Proof. We shall show the contrapositive of the theorem. First we assume that μ is **P**-computable and dominates all other **P**-computable distributions. By Lemma 4.2.7(1), we can assume that μ is further strictly **P**-computable.

Using this μ, we shall construct a **P**-computable set which contains one string on each interval Σ^n, $n \in \mathbb{N}$. We define the function f from $\{0\}^*$ to Σ^* by the following procedure:

begin deterministic algorithm for f
  input 0^n
  if $n = 0$ **then output** λ
  for $k = 1$ **to** n **do**
    (Assume that $a_0 = \lambda$.)
    let $L := \{a_1 a_2 \cdots a_{k-1} 0 \Sigma^{n-k}\}$ and $R = \{a_1 a_2 \cdots a_{k-1} 1 \Sigma^{n-k}\}$
(*) **if** $\hat{\mu}(L) > \hat{\mu}(R)$ **then** let $a_k := 0$ **else** let $a_k := 1$
  end-for
  output $a_1 a_2 \cdots a_n$
end.

To check that f is **P**-computable is easy. By the above algorithm, f satisfies that $|f(0^n)| = n$ for all numbers $n \in \mathbb{N}$. Moreover by line (*), the probability $\hat{\mu}(f(0^n))$ is at most 2^{-n} because at each iteration of the **for**-loop, the probability of the set, either R or L, is reduced by half. To get the desired set, we set $D = f(\{0\}^*)$.

Fix an arbitrary **P**-bi-immune set B in **E**, and let $A = D \cap B$. This set A is infinite because, otherwise, the difference $D - B \cap D$ becomes an infinite subset of $\overline{B}$; this implies that $\overline{B}$ is *not* **P**-immune, a contradiction.

We next show that $(A, \mu) \in \mathrm{Aver}(\mathbf{P}, *)$.

Claim 6 $(A, \mu) \in \mathrm{Aver}(\mathbf{P}, *)$.

Proof of Claim. Since $B \in \mathbf{E}$, there exists a deterministic Turing machine M_0 which computes B in time $2^{cn} + c$ for some positive constant c. To compute A, let us define another machine N as follows:

begin deterministic algorithm for N
  input x (say, $n = |x|$)
  compute $f(0^n)$ in polynomial time
  if $x \neq f(0^n)$ **then reject** and **halt**
  simulate M_0 on input x and **halt**
end.

The machine N actually computes A. Now we must discuss the running time of N on input x. Suppose that $x \notin D$. In this case, N needs polynomial time. Next suppose $x \in D$. In this case,

$$\mathrm{Time}_N(x) \leq c' \cdot (p(|x|) + 2^{c|x|} + c) \leq c'' \cdot 2^{(c+1)|x| - 2\log|x|} \leq \left(\frac{c''}{|x|^2 \cdot \hat{\mu}(x)} \right)^{c+1}.$$

Hence, $\lambda x.\mathrm{Time}_N(x)$ is polynomial on μ-average. ■

Let us define the distribution ν as

$$\hat{\nu}(x) = \begin{cases} \hat{\nu}_{\mathrm{tally}}(0^{|x|}) & \text{if } x \in D, \\ 0 & \text{otherwise.} \end{cases}$$

Because μ avp-dominates ν, Claim 6 implies $(A, \nu) \in \mathrm{Aver}(\mathbf{P}, *)$. To get a contradiction, we must show that $(A, \nu) \notin \mathrm{Aver}(\mathbf{P}, *)$.

Assume that $(A, \nu) \in \mathrm{Aver}(\mathbf{P}, *)$, and we shall derive a contradiction. There exist a deterministic Turing machine M and a polynomial q such that A is computable by M in time q on ν-average. For every string x in A,

$$\mathrm{Time}_M(x) \leq q(|x|/\hat{\nu}_{\mathrm{tally}}(0^{|x|})) \leq q(8|x|(|x|+1)^2).$$

Next we set $p(z) = q(8z(z+1)^2)$ for all z. Using this time bound q, the set A is rewritten as

$$A = \{x \mid M \text{ accepts } A \text{ in time } p(|x|)\}.$$

This yields the $\mathbf{P}$-computability of A. Since $A \subseteq B$, B is *not* $\mathbf{P}$-immune, a contradiction.

This completes the proof. □

Under the assumption $\mathbf{P} = \mathbf{NP}$, Theorem 4.6.2 can be further extended to $O(f)$-universal distributions, where f is any function in the set $o(2^n)$, by a modification of the proof of Lemma 4.1 in [62].

Proposition 4.6.3 **[119]** *Assume that* $\mathbf{P} = \mathbf{NP}$. *For every function* $f \in o(2^n)$, $\mathbf{P}$-comp *has no* $O(f)$-*universal distribution.*

Proof. Assume that $\mathbf{P} = \mathbf{NP}$. Assume that $f \in o(2^n)$, and μ_0 is $O(f)$-universal for $\mathbf{P}$-comp. We note that $g(x) \cdot \hat{\mu}_0(x) \geq \hat{\nu}_{\mathrm{stand}}(x)$ for some $g \in O(f)$ since μ_0 is $O(f)$-universal. Hence, $\hat{\mu}_0(x) \geq \frac{\hat{\nu}_{\mathrm{stand}}(x)}{g(x)} > 2^{-3|x|}$ for almost all x. Let x_{-1} be the minimal string x' such that $\hat{\mu}_0(x) > 2^{-3|x|}$ for all $x \geq x'$.

By definition, there is a polynomial-time Turing machine M which computes μ_0. For each $x \in \Sigma^+$, set $\nu(x) = M(x, 0^{3|x|+4})$ and let $\hat{\nu}(x) = \nu(x) - \nu(x^-)$. In general, ν is not a distribution since $\hat{\nu}$ does not always take a nonnegative value. However, we have $\hat{\nu}(x) > 0$ for all $x \geq x_{-1}$. This is seen as follows: for all x,

$$|\hat{\mu}_0(x) - \hat{\nu}(x)| \leq |\mu_0(x) - \nu(x)| + |\mu_0(x^-) - \nu(x^-)| \leq 2^{-3|x|-4} + 2^{-3|x^-|-4} \leq 2^{-3|x|},$$

and thus, $\hat{\nu}(x) \geq \hat{\mu}_0(x) - 2^{-3|x|} > 0$ if $x \geq x_{-1}$.

Now we define a series of strings $\{x_i \mid i \in \mathbb{N}\}$ as follows. For convenience, write $R(x, y)$ if $y \geq 2^{|x|}$ and $\nu(y) - \nu(x) \geq 2^{|y|} \cdot \hat{\nu}(y)$. Let x_{i+1} be the minimal string such that $R(x_i, x_{i+1})$ holds. This x_{i+1} exists since otherwise, $\hat{\nu}(x_i^+) \leq \nu(y) - \nu(x_i) < 2^{|y|} \cdot \hat{\nu}(y)$ for all y of length $\geq 2^{|x_i|}$, and thus $\hat{\nu}(y) > \frac{1}{c \cdot 2^{|y|}}$ for some constant $c \geq 1$. For each integer $n > |x_i|$, $\sum_{|y|=n} \hat{\nu}(y) > 2^n \cdot \frac{1}{c \cdot 2^n} = \frac{1}{c}$, a contradiction.

The set $\{x_i \mid i \in \mathbb{N}\}$ is expressed by $\{y \mid \exists m < |y| \exists x_0, \ldots, x_m = y \forall i < m[\ x_0 \geq x_{-1}$ and x_{i+1} is the minimal string such that $R(x_i, x_{i+1})]\}$, and hence it belongs to $\mathbf{NP}$. Since $\mathbf{NP}$ collapses to $\mathbf{P}$, $\{x_i \mid i \in \mathbb{N}\}$ is in $\mathbf{P}$. Note that

$$\sum_{i=0}^{\infty} 2^{|x_i|} \cdot \hat{\nu}(x_i) \leq \sum_{i=0}^{\infty} (\nu(x_{i+1}) - \nu(x_i)) \leq \lim_{i \to \infty} \nu(x_i) \leq 1.$$

Let $\hat{\eta}(x) = c \cdot 2^{|x|}(\hat{\nu}(x) + 2^{-3|x|})$ if $x \in \{x_i | i \in \mathbb{N}\}$; otherwise it is 0, where c is an appropriate positive constant. The distribution η is obviously computable in polynomial time, and thus, $\eta \in \mathbf{P}$-comp.

By our definition, for any constant $d > 0$, there exists an i such that

$$\hat{\eta}(x_i) \geq 2^{|x_i|} \cdot (\hat{\nu}(x_i) + 2^{-3|x_i|}) \geq 2^{|x_i|} \cdot \hat{\mu}_0(x_i) \geq d \cdot f(x_i) \cdot \hat{\mu}_0(x_i).$$

This is a contradiction. □

4.7 Domination Relations and Equivalence Relations

As seen in Section 3.4, domination relations can be viewed as an "approximation" or a "reducibility" between two distributions in average-case complexity theory. If two distributions dominate each other, in this paper, we call them "equivalent" since they are close to each other and have almost the same degree of complexity. Equivalence relations were first discussed in [96] using the terminology "approximation within constant factor" to show the closeness of two conditional distributions.

In this section, we shall focus on (average) polynomial-domination and equivalence relations and study their properties.

4.7.1 Condition I

Let us first recall that μ *p-dominates* ν, denoted by $\nu \preceq^{\mathrm{p}} \mu$, if $p(x) \cdot \hat{\mu}(x) \geq \hat{\nu}(x)$, where p is some p-bounded function. Polynomial-domination relations are useful in average-case complexity theory since they do not

change the degree of average running time: namely, provided that μ p-dominates ν, if an algorithm requires polynomial time on μ-average, then this algorithm also runs in polynomial time on ν-average (see Lemma 3.4.6).

Let us consider the following condition:

Condition I. Every distribution in **P**-samp is p-dominated by some distribution in **P**-comp.

The next proposition lists several different conditions which are equivalent to Condition I.

Proposition 4.7.1 *The following conditions are equivalent.*

1. *For every $\mu \in$ **P**-samp, there exists a distribution ν in **P**-comp such that $\mu \preceq^{\mathrm{p}} \nu$.*
2. *For every $\mu \in$ **IP**-samp, there exists a distribution ν in **P**-comp such that $\mu \preceq^{\mathrm{p}} \nu$.*
3. *For every p-honest function $f \in$ **FP** and every $\mu \in$ **P**-comp, there exists a distribution ν in **P**-comp and a p-bounded function p from Σ^* to $\mathbb{R}^+$ such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all strings y.*

Proof. Since **IP**-samp $\subseteq$ **P**-samp, (1) implies (2). By Lemma 3.4.13(1), (2) is equivalent to (3). We show that (2) implies (1). Assume (2). For every $\mu \in$ **P**-samp, take a $\mu' \in$ **IP**-samp which p-dominates μ by Lemma 4.4.8. Use our assumption to obtain a distribution $\nu \in$ **P**-comp which p-dominates μ'. By Lemma 3.4.5(1), we have $\mu \preceq^{\mathrm{p}} \nu$. □

By Theorem 4.5.4, Condition I is derived from the assumption **P** = **PP**. Ben-David, Chor, Goldreich, and Luby [9] further show that if Condition I holds, then no strong one-way function exists. The following is an important fragment of their proof.

Lemma 4.7.2 **[9, 119]** *Assume Condition I. Let f be any function in* **FP**, *let k be a positive number, and let q be any polynomial with $q(n) \geq 1$. Assume that $|x| \leq |f(x)| + k \log |f(x)|$ for almost all x. There exist a set S and a deterministic Turing machine M such that*

(i) $S \subseteq \mathrm{ran}(f)$;

(ii) $\|S \cap \Sigma^n\| < \frac{2^n}{q(n)}$ *for each $n \in \mathbb{N}$; and*

(iii) *M on input x correctly lists all elements of $f^{-1}(x)$ (whenever $f^{-1}(x) = \emptyset$, M outputs 0) in polynomial time unless $x \in S$.*

Proof. Now let $h(\langle w, x\rangle)$ be $\langle y, z\rangle$ if $w \in \{0\}^*$, $|z| = |w|$, $z \sqsubseteq x$, and $f(x) = y$. Note that h is well-defined and p-honest. Take a distribution μ defined as follows: $\hat{\mu}(\langle w, x\rangle) = \hat{\nu}_{\mathrm{stand}}(x) \cdot 2^{-2\mathrm{llog}(|w|)-1}$ if $w \in \{0\}^*$, or else 0. Clearly, $\mu_{h^{-1}} \in$ **IP**-samp $\subseteq$ **P**-samp. Note that $\hat{\nu}_{\mathrm{stand}}(x) \geq \frac{2^{-|x|}}{2(|x|+1)^2}$.

By our assumption Condition I (as well as Proposition 4.7.1(2)), there are an $\eta \in$ **P**-comp and a polynomial r such that $r(|y|) \cdot \hat{\eta}(y, z) \geq \hat{\mu}_{h^{-1}}(y, z)$ for all y and z. So, for each $y \in \mathrm{ran}(f)$, we have $\hat{\eta}(y, z) \geq \frac{2^{-|y|}}{s(|y|)}$ for some z, where $s(n) = 4r(n)n^k(n + n^k + 1)^4$. For each y, let $C_y = \{z \mid \hat{\eta}(y, z) \geq \frac{2^{-|y|}}{s(|y|)}, |z| \leq |y| + k \log |y|\}$.

Define $S = \{y \mid \|C_y\| > q(|y|)s(|y|), y \in \mathrm{ran}(f)\}$. Clearly we have $S \subseteq \mathrm{ran}(f)$. We show that, for almost all n, $\|S^n\| < \frac{2^n}{q(n)}$. Assume otherwise, and let y' be an element of S. Let $n = |y'|$. Then we have

$$\sum_{y \in S^n} \sum_{z \in C_y} \hat{\eta}(\langle y, z \rangle) \geq \frac{\|S^n\| \cdot \|C_{y'}\| \cdot 2^{-n}}{s(n)} > 1,$$

a contradiction. Define a Turing machine M as follows: on input y, by a depth-first search, M computes all elements of C_y if $\|C_y\| \leq q(|y|)s(|y|)$ and lists all elements z of C_y which satisfy $f(z) = y$, or else M outputs 0. Since $f^{-1}(y) \subseteq C_y$, all elements of $f^{-1}(y)$ are printable in polynomial time if $y \in \overline{S}$. This completes the proof. □

Using the hash-function technique of [40] and the amplification technique of probabilistic Turing machines, we can show that Condition I leads to the consequence that every **NP** set is nearly-**RP**.

Theorem 4.7.3 **[119]** *Assume Condition I. Let A be any set in* **NP**. *For every polynomial p with $p(n) \geq 1$ for all $n \in \mathbb{N}$, there exist a set D and a polynomial-time randomized Turing machine M such that $D \subseteq A$, and, for each x, $x \in A - D$ implies $\mathbf{Pr}_M[M(x) \neq A(x)] < \frac{1}{2}$, $x \notin A$ implies $\mathbf{Pr}_M[M(x) \neq A(x)] = 0$, and $\mathbf{Pr}_n[x \in D] < \frac{1}{p(n)}$ for almost all n. Hence, Condition I implies that every* **NP** *set is nearly-***RP**.

Proof. Assume Condition I. Take any set A in **NP** and any polynomial q, and we will show that A satisfies the claim. There exists a set $B \in \mathbf{P}$ such that $A = \{x \mid \exists z \in \Sigma^{|x|}[xz \in B]\}$. Let $B_x = \{z \mid xz \in B\}$ for each x. Assume that there exists an increasing polynomial p such that $\mathbf{Pr}_n[x \in A] \geq \frac{1}{p(n)}$ for almost all n since, otherwise, the theorem is trivial by choosing $D = \emptyset$.

We take the set $H_{n,n+c}$ of hash functions. Let s_k^n be the kth string in the set $\Sigma^{\mathrm{ilog}(n)}$ with respect to the standard order. Define $f(x') = 1xs_k^n hh(y)_{\leftarrow n+c}0^{n-k}$ if $x' = xys_k^n h$ and $y \in B_x$; otherwise $0x'$, where $x \in \Sigma^n$, $h \in H_{n,n+c}$, and $c = \mathrm{ilog}(n)$. Notice that $|x'| \leq |f(x')|$ for all x'. For each x, let $g(x) = \|f^{-1}(x)\|$. For brevity, write $t(n) = 1 + n + \mathrm{ilog}(n) + (n+1)(n + \mathrm{ilog}(n)) + n + \mathrm{ilog}(n)$.

For fixed k, and x of length n, let $\rho_{k,x} = \mathbf{Pr}_{hw}[g(1xs_k^n hw_{\leftarrow n+c}0^{n-k}) = 1 \mid h \in H_{n,n+c}, w \in \Sigma^{n+c}]$. We first show that $\rho_{k,x} \geq \frac{1}{2n+3}$ for almost all n. Now we fix k and x, and assume that $n = |x|$ is sufficiently large and $\mathrm{ilog}(g(x)) \leq k \leq n$. Consider the case $\|B_x\| > 0$. The probability $\rho_{k,x}$ is larger than or equal to the probability over all hw that, for each y in B_x, $h(y)_{\leftarrow k+c} = w_{\leftarrow k+c}$, and h k-distinguishes y on B_x. Thus, we have

$$\rho_{k,x} \geq \|B_x\| \cdot 2^{-(k+c)} \cdot (1 - 2^{-c}) \geq (1 - 2^{-c}) \cdot 2^{-c} \geq \frac{n-1}{2n^2} \geq \frac{1}{2n+3}$$

since $\log n \leq c \leq \log n + 1$. For the case $\|B_x\| = 0$, clearly $\rho_{k,x} = 1$ since $g(1xs_k^n hw_{\leftarrow n+c}0^{n-k}) = 1$ for all k, h, and w. This yields the desired result.

By Lemma 4.7.2, there are a set S and a polynomial-time deterministic Turing machine N which recognizes $\overline{S}$ such that $S \subseteq \mathrm{ran}(f)$ and $\|S \cap \Sigma^{t(n)}\| < \frac{2^{t(n)}}{2(2n+3) \cdot q(n)}$. We define a randomized polynomial-time algorithm M as follows:

begin randomized algorithm M
  input x (say, $n = |x|$)

choose w and h at random ($w \in \Sigma^{n+c}, h \in H_{n,n+c}, c = \mathrm{ilog}(n)$)
let $Result := 0$
for all k $(1 \leq k \leq n)$
run N on $x'_k = 1xs^n_k hw_{\leftarrow k+c}0^{n-k}$
let $Result :=$ OR of $Result$ and $N(x'_k)$
end-for
output $Result$ and **halt**
end.

Let $\delta_{k,x} = \mathbf{Pr}_{hw}[1xs^n_k hw_{\leftarrow k+c}0^{n-k} \in S \mid w \in \Sigma^{n+c}, h \in H_{n,n+c}]$, where $c = \mathrm{ilog}(n)$. Using this $\delta_{k,x}$, we define $D = \{x \in \Sigma^+ \cap A \mid \exists k[\mathrm{ilog}(\|B_x\|) \leq k \leq n \wedge \delta_{k,x} \geq \frac{1}{2(2n+3)} \wedge |x| = n]\}$. We will show that $\mathbf{Pr}_n[x \in D] < \frac{1}{q(n)}$. Assume otherwise. So, we have $\delta_{k,x} \geq \frac{1}{2(2n+3)}$ for some k $(\mathrm{ilog}(\|B_x\|) \leq k \leq n)$ and $x \in D^n$. Since

$$\max\{\delta_{k,x} \mid \mathrm{ilog}(\|B_x\|) \leq k \leq n, x \in \Sigma^n\} \cdot \mathbf{Pr}_n[x \in D] \leq \frac{\|S^{t(n)}\|}{2^{t(n)}} < \frac{1}{2(2n+3)q(n)},$$

we have $\max\{\delta_{k,x} \mid \mathrm{ilog}(\|B_x\|) \leq k \leq n, x \in \Sigma^n\} < \frac{1}{2(2n+3)}$. This is a contradiction. Therefore, $\mathbf{Pr}_n[x \in D] < \frac{1}{q(n)}$.

Now our goal is to prove that (i) $\mathbf{Pr}_M[M(x) = A(x)] \geq \frac{1}{2(2n+3)}$ for all x in $A - D$, and (ii) $\mathbf{Pr}_M[M(x) \neq A(x)] = 0$ for all $x \notin A$. This is enough to establish the theorem because of the worst-case version of the Amplification Lemma. Take any input x of length n. Let $\rho_x = \mathbf{Pr}_{hw}[A(x) = \mathrm{OR}^n_{k=1}N(x'_k)) \mid h \in H_{n,n+c}, w \in \Sigma^{n+c}]$. Note that the probability $\mathbf{Pr}_M[M(x) = A(x)]$ is at least ρ_x. Assume $A(x) = 1$ for a string $x \in \overline{D}$. Note that if $x'_{k'} \in \overline{S}$ and $g(x'_{k'}) = 1$ for some k', then $\mathrm{OR}^n_{k=1}N(x'_k) = 1$. Hence,

$$\rho_x \geq \max\{\rho_{k,x} - \delta_{k,x} \mid \mathrm{ilog}(\|B_x\|) \leq k \leq n\} \geq \frac{1}{2(2n+3)}.$$

For the other case $A(x) = 0$, $N(xs^n_k hw_{\leftarrow n+c}0^{n-k}) = 0$ for all h, w, and k; and thus, $\rho_x = 1$. This completes the proof. □

4.7.2 Condition I′

Polynomial-domination relations are useful but too tight to be considered an effective measure of "approximation" or "reducibility" between distributions in average-case complexity theory. Gurevich [36] later introduced a weaker form of domination relations by requiring a function to be p-bounded "on the average." Following his definition, we also relax Condition I to allow the domination to be, instead, "polynomially-bounded on the average."

Condition I′. Every distribution in **P**-samp is avp-dominated by some distribution in **P**-comp.

Obviously Condition I implies Condition I′, but we suspect that the the converse implication may not be provable affirmatively.

In the case of avp-dominations, we no longer prove a similar equivalence as in Proposition 4.7.1. The following claim is the best possible so far.

Lemma 4.7.4 *The following conditions are equivalent.*

1. *For every $\mu \in \mathbf{IP}_1$-samp, there exists a distribution $\nu \in \mathbf{P}$-comp such that $\mu \preceq^{\mathrm{avp}} \nu$.*
2. *For every p-honest function $f \in \mathbf{FP}$ and every $\mu \in \mathbf{P}$-comp, there exists a $\nu \in \mathbf{P}$-comp and a function p which is polynomial on μ-average such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*

Proof. By Lemma 3.4.13(2). □

To see a consequence of Condition I′, we need a notion of $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and its structural properties. These will appear in Section 7.5.

4.7.3 #P-comp versus P-samp

We first see the gap between the two notions: #**P**-computability and **P**-samplability. The next proposition was shown by Schuler and Watanabe [96] for conditional distributions. We modify their proof to accommodate infinite distributions. To describe the proposition, we need the notion of truth-table samplability of distributions.

Definition 4.7.5 (Truth-Table Samplable Distributions)

1. A distribution μ is $\mathbf{P}^A_{\mathrm{tt}}$-*samplable* if there exist a randomized oracle Turing machine M, a deterministic Turing machine N, and a polynomial p such that
   (i) M with oracle A generates μ in time polynomial in the length of outputs; and
   (ii) on input $(0^i, s)$, N lists all query strings of $Q(M, A, 0^i, s)$ in polynomial-time (without any queries) if s is a code of a computation path given by M with oracle A on input 0^i.
2. The notation $\mathbf{P}^A_{\mathrm{tt}}$-samp denotes the collection of all $\mathbf{P}^A_{\mathrm{tt}}$-samplable distributions, and $\mathbf{P}^{\mathcal{C}}_{\mathrm{tt}}$-samp denotes the union of all distributions in $\mathbf{P}^A_{\mathrm{tt}}$-samp for any $A \in \mathcal{C}$.

Proposition 4.7.6 **[96]** #**P**-comp $\subseteq^{\mathrm{p}}$ $\mathbf{P}^{\mathbf{NP}}_{\mathrm{tt}}$-samp.

Proof. Take an arbitrary #**P**-computable distribution μ. There exist a function $f \in \#\mathbf{P}$ and a polynomial q such that $\left|\hat{\mu}(x) - \frac{f(x,0^i)}{2^{q(|x|,i)}}\right| \leq 2^{-i}$ for all x and $i \in \mathbb{N}$. Without loss of generality, we can assume that this q is increasing. For simplicity, let $\sigma^i_x = \frac{f(x,0^i)}{2^{q(n,i)}}$ for each $i \in \mathbb{N}$. Since $f \in \#\mathbf{P}$, there exists a set $A \in \mathbf{P}$ such that $f(x, 0^i) = \|\{w \in \Sigma^{q(|x|,i)} \mid \langle 0^i, x, w\rangle \in A\}\|$ for all x. For brevity, let $A_i = \{\langle w, x\rangle \mid w \in \Sigma^{q(|x|,i)} \wedge \langle 0^i, x, w\rangle \in A\}$. Let $S^i_n = \{xw \mid \langle x, w\rangle \in A_i \wedge |x| = n \wedge |w| = q(n, i)\}$. Note by Lemma A.4 that $|\sigma^i_x - \sigma^j_x| \leq 2^{-i} + 2^{-j}$ for all integers $i, j > 0$.

We use universal hash functions to approximate the value of $\frac{f(x,0^i)}{2^{q(n,i)}}$. Let us consider the set $H_{n+q(n,i),n+q(n,i)+1}$ of hash functions. We define the following three sets that will be used as oracles:

$$X_0 = \{\langle 0^i, 0^n\rangle \mid \exists x \in \Sigma^n \exists w \in \Sigma^{q(n,i)}[\langle x, w\rangle \in A_i]\};$$

$$X_1 = \{\langle 0^i, 0^n, h, y\rangle \mid \exists xw, x'w' \in S_n^i[xw \neq x'w' \wedge h(xw)_{\leftarrow q(n,i)} = h(x'w')_{\leftarrow q(n,i)} = y_{\leftarrow q(n,i)}]\}; \text{ and}$$

$$X_2 = \{\langle 0^i, 0^n, h, y, 0^j\rangle \mid \exists xw \in S_n^i[h(xw)_{\leftarrow q(n,i)} = y_{\leftarrow q(n,i)} \wedge (xw)_j = 1]\}.$$

It is easy to see that all sets X_0, X_1 and X_2 are in **NP**.

If there exists a string x such that $\mu(x) = 1$, then μ falls into **P**-comp, and thus, the claim is trivial. We therefore assume that there is no such x. Let $p(n) = 2n + 4$. We consider the following randomized algorithm M:

begin randomized algorithm M
  input 0^i
  generate a natural number n_0 at random
  for $n = n_0$ **to** ∞ **do**
    if $\langle 0^{i+p(n)}, 0^n\rangle \notin X_0$ **then** go to $(**)$
    repeat $i + p(n)$ times
      generate h in $H_{m,m+1}$ at random, where $m = n + q(n, i + p(n))$
      generate y in Σ^{m+1} at random
      if $\langle 0^{i+p(n)}, 0^n, h, y\rangle \in X_1$ **then** go to $(*)$
      find the strings $x \in \Sigma^n$ and $w \in \Sigma^{q(n,i+p(n))}$ using X_2 s.t.
        $h(xw)_{\leftarrow r} = y_{\leftarrow r}$ and $\langle x, w\rangle \in A_i$, where $r = q(n, i + p(n))$
      output x and **halt**
$(*)$ **end-repeat**
$(**)$ **end-for**
end.

For each x, we define ${\overline{\rho}_x}^i$ as the probability that the randomized algorithm M outputs x. As in the proof of Theorem 4.4.13, it suffices to show that

(i) $\frac{1-2^{-i-p(n)}}{8(n+1)^2} \cdot \sigma_x^{i+p(n)} \leq {\overline{\rho}_x}^i \leq 2 \cdot \sigma_x^{i+p(n)}$, where $n = |x|$, and

(ii) $|{\overline{\rho}_x}^i - {\overline{\rho}_x}^j| \leq 2^{-i} + 2^{-j}$ for all $i, j > 0$.

To get the desired distribution ν, we set $\hat{\nu}(x) = \lim_{i\to\infty} {\overline{\rho}_x}^i$, and consequently, μ is p-equivalent to ν.

In the following, we shall prove claims (i) and (ii) above. Fix i and n. Let $m = n + q(n, i)$ and $r = q(n, i + p(n))$. Moreover, let $S_x^i = \{w \mid \langle x, w\rangle \in A_{i+p(n)}\}$. First we define $\rho_x^{i,1}$ as the probability, over all $h \in H_{m,m+1}$ and all $y \in \Sigma^{m+1}$, that the algorithm M finds x in a single iteration of the **repeat**-loop; namely,

$$\mathbf{Pr}_{hy}[\langle 0^{i+p(n)}, 0^n, h, y\rangle \notin X_1 \wedge \exists w(h(xw)_{\leftarrow r} = y_{\leftarrow r} \wedge xw \in S_x^i) \mid y \in \Sigma^{m+1}, h \in H_{m,m+1}].$$

This is equivalent to the following:

$$\begin{aligned}\rho_x^{i,1} &= \sum_{w\in S_x^i} \mathbf{Pr}_{hy}[h(xw)_{\leftarrow r} = y_{\leftarrow r} \wedge \\ &\qquad \wedge \forall z \in S_x^i - \{xw\}(h(z)_{\leftarrow r} \neq h(xw)_{\leftarrow r}) \mid y \in \Sigma^{m+1}, h \in H_{m,m+1}] \\ &= \sum_{w\in S_x^i} \mathbf{Pr}_{hy}[h(xw)_{\leftarrow r} = y_{\leftarrow r} \wedge h \text{ } m\text{-distinguishes } xw \text{ on } S_n^i \mid y \in \Sigma^{m+1}, h \in H_{m,m+1}].\end{aligned}$$

Hence, by Proposition 2.6.2(2), we have (letting $c = 1$ in the proposition)

$$\frac{1}{2}\cdot \sum_{w\in S_x^i} 2^{-q(n,i+p(n))} = \sum_{w\in S_x^i} 2^{-q(n,i+p(n))}\cdot(1-2^{-1}) \leq \rho_x^{i,1} \leq \sum_{w\in S_x^i} 2^{-q(n,i+p(n))}.$$

Since $\|S_x^i\| = f(x, 0^{i+p(n)})$, we have

$$\frac{1}{2}\cdot \sigma_x^{i+p(n)} = \frac{1}{2}\cdot \frac{f(x,0^{i+p(n)})}{2^{q(n,i+p(n))}} \leq \rho_x^{i,1} \leq \frac{f(x,0^{i+p(n)})}{2^{q(n,i+p(n))}} = \sigma_x^{i+p(n)}.$$

Next we define ρ_x^i as the probability, over all $h \in H_{m,m+1}$ and all $y \in \Sigma^{m+1}$, that the algorithm M finds x during $i + p(n)$ iterations of the **repeat**-loop. We fist estimate the error probability. The error probability is at most $2^{-i-p(n)}$. As a consequence, we have

$$\left(1 - \frac{1}{2^{i+p(n)}}\right)\cdot \sigma_x^{i+p(n)} < \rho_x^i \leq \sigma_x^{i+p(n)}.$$

As in Theorem 4.4.13, we can show that $|\rho_x^i - \rho_x^j| \leq 2^{-p(n)+1}(2^{-i} + 2^{-j})$.

The following analysis of the algorithm is similar to the proof of Theorem 4.4.13.

Let $\tau_n^i = \sum_{x:|x|=n} \rho_x^i$. The probability $\overline{\rho}_x{}^i$ is equivalent to the probability that, for every $k \leq n$, k is first generated, and the algorithm fails to halt until it reaches stage n of the **for**-loop and finds x. For each k, with probability $\frac{1}{2^{2\mathrm{llog}(k)+1}}$, we generate integer k. Following the algorithm, we eventually reach n and with probability ρ_x^i, we output x. Hence, if $0 \leq k < n$, then the probability that x is actually output is $2^{-2\mathrm{llog}(k)-1}\cdot\rho_x^i\cdot\prod_{j=k}^{n-1}(1-\tau_j^i)$, and if $k = n$, then this probability is exactly $2^{-2\mathrm{llog}(n)-1}\cdot\rho_x^i$. Thus,

$$\overline{\rho}_x{}^i = \sum_{k=0}^{n-1}\left(\frac{\rho_x^i}{2^{2\mathrm{llog}(k)+1}}\prod_{j=k}^{n-1}(1-\tau_j^i)\right) + \frac{\rho_x^i}{2^{2\mathrm{llog}(n)+1}}.$$

Hence, we have

$$\frac{1-2^{-i-p(n)}}{8(n+1)^2}\cdot\sigma_x^{i+p(n)} \leq \frac{1}{2^{2\mathrm{llog}(n)+1}}\rho_x^i \leq \overline{\rho}_x{}^i \leq \rho_x^i\cdot\sum_{k=0}^{n}\frac{1}{2^{2\mathrm{llog}(k)+1}} < \frac{\pi^2}{6}\rho_x^i < 2\cdot\sigma_x^{i+p(n)}.$$

For the condition (ii), we follow the proof of Proposition 4.4.13. □

There may be possible to replace the symbol "$\subseteq^{\mathrm{p}}$" in the above proposition by "$\subseteq$." However, because our algorithm needs to know the length of output strings x before the actual simulation of $\hat{\mu}(x)$, we cannot conclude that $\#\mathbf{P}\text{-comp} \subseteq \mathbf{P}_{\mathrm{tt}}^{\mathbf{NP}}\text{-samp}$.

Here is a corollary of Proposition 4.7.6. We note that $\mathbf{NP} \subseteq \mathbf{BPP}$ if and only if $\mathbf{NP} = \mathbf{RP}$ [52].

Corollary 4.7.7 $\mathbf{NP} \subseteq \mathbf{BPP}$ *implies* $\#\mathbf{P}$-comp $\cong^{\mathrm{p}}$ $\mathbf{P}$-samp.

Proof. We note that $\mathbf{P}$-samp $\subseteq$ $\#\mathbf{P}$-comp. It suffices to show that $\#\mathbf{P}$-comp $\subseteq^{\mathrm{p}}$ $\mathbf{P}$-samp under the assumption $\mathbf{NP} \subseteq \mathbf{BPP}$. Let us assume $\mathbf{NP} \subseteq \mathbf{BPP}$. Take an arbitrary distribution μ in $\#\mathbf{P}$-comp. By Proposition 4.7.6, there is a $\nu \in \mathbf{P}_{\mathrm{tt}}^{\mathbf{NP}}$-samp such that μ is p-equivalent to ν. Under our assumption, ν belongs to $\mathbf{P}^{\mathbf{BPP}}$-samp. Using Proposition 4.4.13, there is a distribution ξ in $\mathbf{P}$-samp such that $\nu \approx^{\mathrm{p}} \xi$. Hence, $\mu \approx^{\mathrm{p}} \xi$. □

4.7.4 Condition II′

In this subsection, we study the following condition on equivalence relations:

Condition II′. $\mathbf{P}$-samp $\subseteq^{\mathrm{avp}}$ $\mathbf{P}$-comp.

Clearly Condition II′ implies Condition I′.

As in the previous subsection, we can show the following proposition.

Proposition 4.7.8 *The following two conditions are equivalent.*

1. $\mathbf{IP}_1$-samp $\subseteq^{\mathrm{avp}}$ $\mathbf{P}$-comp.
2. *For every p-honest* $f \in \mathbf{FP}$ *and every* $\mu \in \mathbf{P}$-comp, *there exists* $\nu \in \mathbf{P}$-comp *and functions* p, q *which are polynomial on* μ*-average such that* $\sum_{x \in f^{-1}(y)} q(x)\hat{\mu}(x) \geq \hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ *for all* y.

Proof. By Lemma 3.4.14(2). □

In what follows, we shall show that $\mathbf{IP}_1$-samp $\subseteq^{\mathrm{avp}}$ $\mathbf{P}$-comp implies $\mathbf{P} = \mathbf{RP}$, and thus if Condition II′ is true, then $\mathbf{RP}$ collapses to $\mathbf{P}$. In the following proof, the worst-case version of the Amplification Lemma to one-sided bounded-error probabilistic algorithms is effectively used to make its error probability exponentially small.

Proposition 4.7.9 **[119]** $\mathbf{IP}_1$-samp $\subseteq^{\mathrm{avp}}$ $\mathbf{P}$-comp *implies* $\mathbf{P} = \mathbf{RP}$.

Proof. Consider an arbitrary $A \in \mathbf{RP}$. We prove that A belongs to $\mathbf{P}$. By the amplification lemma [91], there is a strictly increasing polynomial p and a set $B \in \mathbf{P}$ such that, for every $x \in \Sigma^n$, $\mathbf{Pr}_y[\langle x, y\rangle \notin B \mid y \in \Sigma^{p(n)}] \leq 2^{-n}$ if $x \in A$, and otherwise, $\mathbf{Pr}_y[\langle x, y\rangle \in B \mid y \in \Sigma^{p(n)}] = 0$.

Let μ be the distribution defined by $\hat{\mu}(xy) = \hat{\nu}_{\mathrm{stand}}(x) \cdot 2^{-p(|x|)}$ if $|y| = p(|x|)$, or else $\hat{\mu}(xy) = 0$. Clearly μ is $\mathbf{P}$-computable. Let

$$f(xy) = \begin{cases} x1^{p(|x|)} & \text{if } |y| = p(|x|) \text{ and } \langle x, y\rangle \in B, \\ x0^{p(|x|)} & \text{if } |y| = p(|x|) \text{ and } \langle x, y\rangle \notin B, \\ xy & \text{otherwise.} \end{cases}$$

By our assumption and Proposition 4.7.8, we have a distribution $\nu \in \mathbf{P}$-comp and a function q which is polynomial on μ-average such that $\sum_{x \in f^{-1}(y)} q(x)\hat{\mu}(x) \geq \hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{q(x)}$ for all y. Since $\sum_{x \neq \lambda} \frac{q(x)^{1/k}}{|x|}\hat{\mu}(x) \leq c$ for some constants $k, c \geq 1$, we have $q(x) \leq \left(\frac{c \cdot |x|}{\hat{\mu}(x)}\right)^k$ for all nonempty strings x with $\hat{\mu}(x) > 0$. Thus, for almost all x and all y of length $p(|x|)$,

$$q(xy) \leq \left(\frac{c \cdot |xy|}{\hat{\mu}(xy)}\right)^k \leq \left(c(|x| + p(|x|))(|x|+1)2^{|x|+p(|x|)}\right)^k \leq \left(2^{|x|} \cdot 2^{|x|+p(|x|)}\right)^k = 2^{r(|x|)},$$

where $r(n) = (2n + p(n))^k$. Since $\nu \in \mathbf{P}$-comp, there exists a deterministic polynomial-time Turing machine M such that $|\hat{\nu}(x) - M(x, 0^i)| < 2^{-i}$. Let $M'(x) = M(x, 0^{r(|x|)+2|x|})$. By definition, $|\hat{\nu}(x) - M'(x)| < 2^{-r(|x|)-2|x|}$ for all x.

Let $x \in \Sigma^n$. Assume that $x \in A$. Then, we have

$$\hat{\nu}(x1^{p(n)}) \geq \sum_{z \in f^{-1}(x1^{p(n)})} \frac{\hat{\mu}(z)}{q(z)} \geq \frac{\|f^{-1}(x1^{p(n)})\|}{2^{r(n)}} \cdot \frac{\hat{\nu}_{\mathrm{stand}}(x)}{2^{p(n)}} \geq \frac{2^n - 1}{2^{r(n)+2n}}$$

since $\hat{\nu}_{\mathrm{stand}}(x) \geq \frac{1}{2n^2 \cdot 2^n} \geq \frac{1}{2^{2n}}$ if $n \geq 7$. Hence, $M'(x) > \hat{\nu}(x1^{p(n)}) - 2^{-r(n)-2n} \geq 2^{-r(n)-2n}(2^n - 2)$. In the case that $x \notin A$, $\hat{\nu}(x1^{p(n)}) \leq \sum_{z \in f^{-1}(x1^{p(n)})} q(z) \cdot \hat{\mu}(z) = 0$. Hence, $M'(x) < \hat{\nu}(x1^{p(n)}) + 2^{-r(n)-2n} = 2^{-r(n)-2n}$. Now we have a complete characterization of A in terms of M'; namely, $A \cap \Sigma^n = \{x \in \Sigma^n \mid M'(x) \geq 2^{-r(n)-2n}(2^n - 2)\}$ for almost all n. Since M' halts in polynomial time, A is also computable in polynomial time. □

The above proposition does not suffice to imply that $\mathbf{P} = \mathbf{PP}$, since the worst-case version of the Amplification Lemma may not hold for $\mathbf{PP}$ sets.

4.7.5 Condition II

In contrast to Condition II′, we shall consider its polynomial version, which leads to the conclusion that $\mathbf{P} = \mathbf{NP}$. Formally we define Condition II as follows:

Condition II. $\mathbf{P}$-samp $\subseteq^{\mathrm{p}}$ $\mathbf{P}$-comp.

Note that Condition II implies Condition II′ as well as Condition I and that, by Theorem 4.5.4, Condition II is true if $\mathbf{FP} = \#\mathbf{P}$.

We start with the following proposition.

Proposition 4.7.10 **[119]** *The following three conditions are equivalent.*

1. $\mathbf{P}$-samp $\subseteq^{\mathrm{p}}$ $\mathbf{P}$-comp.
2. $\mathbf{IP}$-samp $\subseteq^{\mathrm{p}}$ $\mathbf{P}$-comp.
3. *For every p-honest $f \in \mathbf{FP}$ and every $\mu \in \mathbf{P}$-comp, there exist a distribution ν in $\mathbf{P}$-comp and p-bounded functions p and q from Σ^* to $\mathbb{R}^+$ such that $\sum_{x \in f^{-1}(y)} q(x)\hat{\mu}(x) \geq \hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y.*

Proof. By Lemma 3.4.14(1). □

By Proposition 4.7.10, we can replace **P**-samp in Condition II by a smaller set **IP**-samp. At the end of this section, we shall see that we also can replace **P**-samp by a larger set #**P**-comp.

Next we shall prove that Condition II yields the consequence that **NP** collapses to **P**. We first strengthen Lemma 4.7.2 under the assumption that **P**-samp is p-included in **P**-comp.

Lemma 4.7.11 **[119]** *Assume that* **P**-samp $\subseteq^{\mathrm{p}}$ **P**-comp. *For any set* $B \in \mathbf{P}$ *and any polynomial* p, *let* $S_B = \{x \mid \|B_x\| \leq p(|x|)\}$, *where* $B_x = \{z \in \Sigma^{|x|} \mid xz \in B\}$. *There exists a deterministic Turing machine* M *such that, for each* $n \in \mathbb{N}$, M *on input* x *in* $S_B \cap \Sigma^n$ *lists all elements of* B_x *(whenever* $B_x = \emptyset$, M *outputs* 0*) in polynomial time.*

Proof. Assume that **P**-samp $\subseteq^{\mathrm{p}}$ **P**-comp. We define a p-honest function f as follows: $f(\langle w, yx\rangle) = \langle 1y, z\rangle$ if $w = s_k^{|xy|}$ for some k, $|x| = |y|$, $|z| = k$, $z \sqsubseteq x$, and $yx \in B$; otherwise, $\langle 0y, x\rangle$. Let $\hat{\mu}(\langle w, x\rangle) = \hat{\nu}_{\mathrm{stand}}(x) \cdot 2^{-2\mathrm{llog}^2(|x|)-1}$ if $w \in \Sigma^{\mathrm{llog}(|x|)}$, or else 0, where $\mathrm{llog}^2(n) = \mathrm{llog} \circ \mathrm{llog}(n)$.

Since $\mu_{f^{-1}} \in \mathbf{P}$-samp, our assumption ensures that there are an $\eta \in \mathbf{P}$-comp and a polynomial r such that $r(|y|+|z|) \cdot \hat{\mu}_{f^{-1}}(\langle y, z\rangle) \geq \hat{\eta}(\langle y, z\rangle) \geq \hat{\mu}_{f^{-1}}(\langle y, z\rangle)/r(|y|+|z|)$ for all y and z. Denote by $D_{y,z}$ the collection of x such that $z \sqsubseteq x$ and $x \in B_y$. If $y \in S_B$, then $\|D_{y,z}\| \leq p(|y|)$. Note that $\hat{\mu}_{f^{-1}}(\langle y, z\rangle) = \frac{\|D_{y,z}\|}{2^{2\mathrm{llog}^2(|y|)+1}} \cdot \frac{2^{-2|y|}}{2^{2\mathrm{llog}(2|y|)+1}}$. For simplicity, let $q(n)$ be $2\mathrm{llog}^2(n) + 2\mathrm{llog}(2n) + 2$. Hence, $\hat{\eta}(\langle 1y, z\rangle) \geq \frac{2^{-2|y|}}{r(|y|) \cdot 2^{q(|y|)}}$ if $D_{y,z} \neq \emptyset$.

Let N be a polynomial-time Turing machine which computes $\hat{\eta}$. Let d be the minimal positive integer such that $3 \cdot r(2n) \cdot 2^{q(n)} \leq 2^{d \cdot \mathrm{llog}(n)}$ for almost all n. We define a new machine N' as $N'(\langle y, z\rangle) = N(\langle y, z\rangle, 0^{2|y|+d\cdot\mathrm{llog}(|y|)+1})$. Hence, $N'(\langle 1y, z\rangle) > \hat{\eta}(\langle 1y, z\rangle) - 2^{-2|y|-d\cdot\mathrm{llog}(|y|)-1} > \frac{2 \cdot 2^{-2|y|}}{2^{d\cdot\mathrm{llog}(|y|)}}$. For each $y \in \Sigma^n$, let $C_y = \{z \mid N'(\langle 1y, z\rangle) > \frac{2 \cdot 2^{-2n}}{2^{d\cdot\mathrm{llog}(n)}}, |z| \leq n\}$.

We note that if $z \in C_y$, then $\hat{\eta}(\langle 1y, z\rangle) > \frac{2^{-2|y|}}{2^{d\cdot\mathrm{llog}(n)}}$. For each $n > 0$ and any $y \in S_B \cap \Sigma^n$,

$$\frac{r(2n)^2 \cdot 2^{-2n}}{2^{q(n)}} \cdot \sum_z \|D_{y,z}\| \geq \sum_z r(|y|+|z|)\mu_{f^{-1}}(\langle 1y, z\rangle) \geq \sum_z \hat{\eta}(\langle 1y, z\rangle) \geq \frac{2^{-2n}}{2^{d\cdot\mathrm{llog}(n)}}\|C_y\|.$$

Therefore, $\|C_y\| \leq 2^{d\cdot\mathrm{llog}(n)-q(n)} \cdot r(2n)^2 \cdot \sum_z \|D_{y,z}\| \leq 2^d r(2n)^2 p(n)^3 n$.

Note that $B_y \subseteq C_y$ if $y \in S_B$. Since C_y is printable in polynomial time, all elements of B_y are printable in polynomial time. This completes the proof. □

Theorem 4.7.12 **[119]** **P**-samp $\subseteq^{\mathrm{p}}$ **P**-comp *implies* **P** = **NP**.

Proof. Let us assume that **P**-samp $\subseteq^{\mathrm{p}}$ **P**-comp. Let A be an arbitrary set in **NP**. We shall show that $A \in \mathbf{RP}$ since $\mathbf{P} = \mathbf{RP}$ by Proposition 4.7.9. It suffices to consider a set A of the form $A = \{x \mid \exists z \in \Sigma^{|x|}[xz \in B]\}$ for some $B \in \mathbf{P}$. Let $B_x = \{z \in \Sigma^{|x|} \mid xz \in B\}$.

Let us define

$$\hat{B} = \{x'z' \mid \exists k h x z[x, z \in \Sigma^n \wedge x' = x s_k^n h h(z)_{\leftarrow k+c} 0^{n-k}$$

$$\wedge z' = z10^{|x'|-|z|} \wedge xz \in B \wedge h \in H_{n,n+c} \wedge c = \mathrm{ilog}(n)]\}.$$

Since $B \in \mathbf{P}$, $\hat{B}$ is also in $\mathbf{P}$. Let $S_{\hat{B}} = \{x' \mid \|\hat{B}_{x'}\| \leq 1\}$, where $\hat{B}_{x'} = \{z' \in \Sigma^{|x'|} \mid x'z' \in \hat{B}\}$.

We define $\rho_{k,x} = \mathbf{Pr}_{hw}[xs_k^n hw_{\leftarrow k+c}0^{n-k} \in S_{\hat{B}} = 1 \mid h \in H_{n,n+c} \wedge w \in \Sigma^{n+c}]$. We first show that, for almost all n and all x with $\|B_x\| > 0$, $\rho_{k,x} > \frac{1}{2n+3}$. Now fix k and x and assume that $n = |x|$ is sufficiently large, and $\mathrm{ilog}(\|B_x\|) \leq k \leq n$. The probability $\rho_{k,x}$ is at most the sum of the probabilities over all strings hw that, for each $z \in B_x$, $h(z)_{\leftarrow k+c} = w_{\leftarrow k+c}$, and h k-distinguishes z on B_x. Thus, we have

$$\rho_{k,x} > \|B_x\| \cdot (1 - 2^{-c}) \cdot 2^{-(k+c)} \geq (1 - 2^{-c})2^{-c} = \frac{n-1}{2n^2} \geq \frac{1}{2n+3}.$$

We apply Lemma 4.7.11 to the set $S_{\hat{B}}$, and we obtain a polynomial-time deterministic Turing machine N which recognizes $S_{\hat{B}}$. We define a randomized polynomial-time algorithm M as follows:

begin randomized algorithm M
  input x (say, $n = |x|$)
  choose w, h at random ($w \in \Sigma^n, h \in H_{n,n+c}, c = \mathrm{ilog}(n)$)
  let *Result* $= 0$
  for all k $(1 \leq k \leq n)$
    run N on $x'_k = xs_k^n h_j w$
    let *Result* $=$ OR of *Result* and $N(x'_k)$
  end-for
  output *Result*
end.

Our goal is to prove that $\mathbf{Pr}_M[M(x) = A(x)] \geq \frac{1}{2(2n+3)}$ for almost all x. Take any input x of length n. Let $\rho_x = \mathbf{Pr}_{hw}[A(x) = \mathrm{OR}_{k=1}^n N(x'_k)) \mid h \in H_{n,n+c}, w \in \Sigma^n]$. Note that the probability $\mathbf{Pr}_M[M(x) = A(x)]$ is at least ρ_x.

Assume $A(x) = 1$. Note that if $0 < g_{\hat{B}}(x'_{k'}) \leq 1$ for some k', then $\mathrm{OR}_{k=1}^n N(x'_k) = 1$. The probability $\rho_x = \mathbf{Pr}_{hw}[A(x) = \mathrm{OR}_{k=1}^n N(x'_k) \mid h \in H_{n,n+c}, w \in \Sigma^n]$ is at least the sum of the probability over all hw that, for each $z \in B_x$ and for some k with $\mathrm{ilog}(g_B(x)) \leq k \leq n$, $h(z)_{\leftarrow k+c} = w_{\leftarrow k+c}$ and h k-distinguishes z on B_x. Hence, $\rho_x > g_B(x) \cdot (1 - 2^{-c}) \cdot 2^{-(k+c)} \geq (1 - 2^{-c})2^{-c} \geq \frac{1}{2n+5}$. For the other case $A(x) = 0$, $N(xs_k^n hw_{\leftarrow k+c}0^{n-k}) = 0$ for all h, w, and k; thus $\rho_x = 1$. This completes the proof. □

Corollary 4.7.13 **[119]** $\mathbf{P}$-samp $\subseteq^{\mathrm{p}}$ $\mathbf{P}$-comp *if and only if* $\#\mathbf{P}$-comp $\subseteq^{\mathrm{p}}$ $\mathbf{P}$-comp.

Proof. (If – part) This is obvious since $\#\mathbf{P}$-comp $\supseteq$ $\mathbf{P}$-samp by Proposition 4.5.5.

(Only if – part) Assume Condition II. By Theorem 4.7.12, we have $\mathbf{P} = \mathbf{NP}$. In particular, $\mathbf{NP} \subseteq \mathbf{BPP}$. By Corollary 4.7.7, every $\#\mathbf{P}$-computable distribution is p-equivalent to some distribution which can be sampled by a randomized Turing machine in time polynomial in its output. □

4.8 Other Topics

There are several intriguing distributional issues which are not discussed in this chapter. Here we present two different approaches.

Ranking Distributions. Reischuk and Schindelhauer [84] introduced a new type of distributions that allows precise complexity classification of distributional problems. They called such distributions *rankable distributions*.

Given a distribution μ, its *ranking function* $\mathrm{rank}_\mu(x)$ is defined as $\mathrm{rank}_\mu(x) = \|\{z \mid \hat{\mu}(z) \geq \hat{\mu}(x)\}\|$. A distribution μ is *polynomially rankable* (**P**-rankable, for short) if the function rank_μ is one-one and **P**-computable [84]. Let **P**-rank denote the collection of all **P**-rankable distributions. Note that, for every x, $\mathrm{rank}_\mu(x) \cdot \hat{\mu}(x) \leq \sum_z \hat{\mu}(z) = 1$. This implies that $\mathrm{rank}_\mu(x) \leq \frac{1}{\hat{\mu}(x)}$ unless $\hat{\mu}(x) = 0$. In particular, if μ is positive and supportive, $\log(\mathrm{rank}_\mu(x)) \leq p(|x|)$ holds for some polynomial p.

For simplicity, let t be a strictly increasing function on $\mathbb{R}^+$. A function f from Σ^* to $\mathbb{N}$ is called *t-average with respect to* rank_μ if $\sum_{x:\mathrm{rank}_\mu(x)\leq m} \frac{t^{-1}(f(x))}{|x|} \leq m$ for any number $m \geq 0$. We say that a distribution (D, rank_μ) is solvable in *average polynomial time with respect to rankability* if there exist a polynomial p and a deterministic Turing machine M computing D such that $\lambda x.\mathrm{Time}_M(x)$ is p-average with respect to rank_μ.

Watanabe [114] pointed out the following relationship between **P**-computable distributions and **P**-rankable distributions. Take a distribution μ satisfying $\log(\mathrm{rank}_\mu(x)) \leq |x|^d$ for some constant $d > 0$. Let us then define

$$\hat{\mu}_*(x) = \frac{c_0}{\mathrm{rank}_\mu(x) \cdot \log^2(\mathrm{rank}_\mu(x))},$$

where c_0 is the normalizing constant and $\log^2 z = (\log z)^2$.

Proposition 4.8.1 **[114]** *Let μ be a distribution and assume that $\log(\mathrm{rank}_\mu(x)) \leq |x|^k$ for some $k > 0$. For any function f from Σ^* to $\mathbb{N}$, if f is polynomial on μ_*-average, then f is polynomial-average with respect to rank_μ.*

Proof Sketch. Assume that f is polynomial on μ_*-average. Then we can take constants $c, k > 0$ such that, for all $m > 0$,

$$\sum_{x:\mathrm{rank}_\mu(x)\leq 2^m} \frac{f(x)^{1/k}}{|x|} \leq c \cdot m^2 \cdot 2^m.$$

For any m, let X_m be the set of strings x such that $2^{m-1} < \mathrm{rank}_\mu(x) \leq 2^m$. Now we can claim that there exists a constant $e > 0$ such that, for any sufficiently large m,

$$\sum_{x\in X_m} \frac{f(x)^{1/16dk}}{|x|} \leq 2^{m-2}.$$

To show the claim, let $X'_m = \{x \in X_m \mid f(x)^{1/k}/|x| \leq m^4\}$ and $X''_m = \{x \in X_m \mid f(x)^{1/k}/|x| > m^4\}$. For the set X'_m, it holds that

$$\sum_{x\in X'_m} \frac{f(x)^{1/16dk}}{|x|} \leq \frac{2^m}{m^{1/4t-1/8dt^2}} \leq 2^{m-3}.$$

On the other hand, the set X''_m satisfies that

$$\sum_{x \in X'''_m} \frac{f(x)^{1/16dk}}{|x|} \leq \frac{cm^2 2^m}{m^{4(1-1/16t)}} \leq 2^{m-3}.$$

Combining the above two sums, we obtain the desired result.

Therefore, for any sufficiently large number m, $\sum_{x:\mathrm{rank}_\mu(x) \leq 2^m} \frac{f(x)^{1/16dk}}{|x|} \leq 2^m$, which implies that f is p-average with respect to rank_μ. ■

We note that Belanger and Wang [6] show that if RBTP is in $\mathrm{Aver}(\mathbf{P}, *)$, then any problem in $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-rank})$ is solvable in average polynomial time with respect to rankability.

Indistinguishability of Distributions. Another topic is the complexity of distributions. We shall follow Meyer's terminology [73]. Two distributions μ and ν are called "statistically indistinguishable" if $\lim_{n\to\infty} n^k \sum_{x:|x|=n} |\hat{\mu}_n(x) - \hat{\nu}_n(x)| = 0$ for all $k \in \mathbb{N}$. Similarly, μ and ν are "circuit indistinguishable" if, for all families of polynomial-size circuits $\mathcal{C} = \{C_n\}_{n\mathbb{N}}$, $\lim_{n\to\infty} n^k |\sum_{x:|x|=n} C_n(x)(\hat{\mu}_n(x) - \hat{\nu}_n(x))| = 0$ for all $k \in \mathbb{N}$; and "algorithmically indistinguishable" if, for all polynomial-time probabilistic Turing machines M, $\lim_{n\to\infty} n^k |\sum_{x:|x|=n} \mathbf{Pr}_M[M(x) = 1] \cdot (\hat{\mu}_n(x) - \hat{\nu}_n(x))| = 0$ for all $k \in \mathbb{N}$.

Meyer [73] show the following separation result:

Theorem 4.8.2 **[73]** *Let f be a space-constructible function on $\mathbb{N}$ such that $\lim_{n\to\infty} \frac{n^k}{f(n)} = 0$ for all $k \in \mathbb{N}$. There exist two $O(f^2)$-space samplable distributions μ and ν such that μ and ν are algorithmically indistinguishable but circuit distinguishable.*

We do not intend to include any proof of this theorem. The interested reader may refer to [73].

Chapter 5

Average Polynomial Time Reducibilities

5.1 Introduction

In the early 1970's, Cook [22], Karp [49], and Levin [59] took a pioneering step towards the classification of the hardest problems in **NP**. They introduced notions of resource-bounded reducibilities among **NP** problems. The idea of these reducibilities is as follows: a problem T is recognized to be at least as *hard* as another problem S if the problem S can be transformed into the problem T in polynomial time so that if T is solved easily, then so is S. The problems "hardest" in this sense among **NP** problems are called **NP**-*complete*. Typical **NP**-complete problems are the *satisfiability problem*, *Hamiltonian circuit problem*, and *traveling salesman problem*. For more **NP**-complete problems, see [26].

In average-case complexity theory, Levin has introduced a similar notion of polynomial-time many-one reducibility among distributional decision problems and showed that the randomized bounded tiling problem is one of the hardest problems in Dist(**NP**, **P**-comp). His reduction from a distributional problem (A, μ) to another distributional problem (B, ν) requires a polynomial-time many-one reduction that maps set A to set B and employs a so-called *domination condition* between μ and ν. This domination condition guarantees that instances occurring with high probability are mapped by the reduction to instances occurring with high probability. This requirement is essential in Levin's theory of average-case **NP**-completeness in order to ensure the closure property of a class under these reductions. At the same time, it makes completeness proofs of given distributional problems difficult to achieve.

Since Levin's discovery of average-case **NP**-completeness, researchers have introduced several interesting notions of reducibilities into the theory of average-case **NP**-completeness. We will review these reducibilities in this chapter.

Section 5.2 will introduce two of the most important reducibilities among distributional decision problems: deterministic many-one reducibility and Turing reducibility. Levin [60] introduced a notion of many-one

reducibility, and the notions of deterministic Turing reducibility and random many-one reducibility were introduced and studied in [60, 9, 44, 12, 96]. Other reductions of interest are logspace many-one reductions [9] and logspace many-one reductions which are p-honest [32].

Section 5.2 will introduce *deterministic many-one reducibility* and *polynomial-time isomorphism* among distributional decision problems, and then the notion of *deterministic Turing reducibility* will be introduced. The Turing reducibility will be particularly used to build the average polynomial-time hierarchy in Chapter 6.

In Section 5.3, we shall exhibit several many-one complete problems for Dist($\mathbf{NP}, \mathbf{P}$-comp). We first show that the *randomized bounded halting problem* is complete for Dist($\mathbf{NP}, \mathbf{P}$-comp), and then the other distributional problems are shown to be polynomially isomorphic to this problem.

In Section 5.4, we shall discuss several incompleteness results. The choice of distributions is important when one attempts to prove that some distributional problem is complete for Dist($\mathbf{NP}, \mathbf{P}$-comp). Some distributions are such that their associated distributional problems cannot complete for Dist($\mathbf{NP}, \mathbf{P}$-comp). For example, flat distributions and sparse distributions have this characteristic.

Flat distributions are often arise in graph-related decision problems; however, no distributional problems with flat distributions become complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) unless $\mathbf{EXP} = \mathbf{NEXP}$. Venkatesan and Levin [106] proposed new reduction which uses random coin tosses. In Section 5.5, we shall introduce probabilistic reducibility, called *bounded-error probabilistic truth-table reducibility*.

In Section 5.6, some structural results are shown.

Major Contributions. Some results in Section 5.6 are from Schuler and Yamakami [97], and Section 5.5 provides a series of new results. More precisely, the following are the major contributions to this chapter.

Theorem 5.4.5 shows that sparse distributions fail to make the corresponding distributional problems p-m-complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) unless $\mathbf{P} = \mathbf{NP}$.

Lemma 5.5.5 is a relativized version of the Amplification Lemma (Lemma 3.5.31).

Lemma 5.5.4 shows that skew avbpp-tt-reducibility is closed under p-m-reductions; namely, if (A, μ) is p-m-reducible to some distributional problem which is skew avbpp-tt-reducible to (B, ν), then (A, μ) is skew avbpp-tt-reducible to (B, ν).

Lemma 5.5.2 shows that, if (A, μ) is skew avbpp-tt-reducible to (B, ν), then there exists a problem (B', ν') which is p-m-reducible to (B, ν) such that (A, μ) is skew avbpp-tt-reducible to (B', ν') via a reduction machine which queries strings of length greater than the input size.

Proposition 5.6.5 shows that, for every recursive set D not in $\mathbf{P}$, there exists an incomparable pairs with respect to $\leq_T^{\mathrm{p}}$.

Theorem 5.5.9 shows that the distributional bounded halting problem with a flat distribution is bpp-tt-complete for Dist($\mathbf{NP}, \mathbf{P}$-comp).

Proposition 5.5.7 introduces a new result regarding the transitivity of avbpp-tt- and bpp-tt-reducibilities.

5.2 Deterministic Reducibility

Reducibility is one of the most important tools in computational complexity theory for assessing the relative complexity of two given problems. This section will present the formal definitions of deterministic reducibilities among distributional decision problems and develop their basic structural properties.

5.2.1 Many-One Reducibility

In this section, we shall formally introduce two notions of many-one reducibilities: *polynomial-time many-one* and *average polynomial-time many-one* reducibilities.

A many-one reducibility among distributional decision problems requires mappings among sets and constraints among distributions. Let us consider the worst-case polynomial-time many-one reduction f between two sets A and B. The reduction ensures the relationship $A = \{x \mid f(x) \in B\}$. If B is computed by a machine M in polynomial time, then a simple algorithm N which first computes $f(x)$ and then simulates machine M on input $f(x)$ actually witnesses the **P**-computability of A. Now we switch from sets A and B to two distributional decision problems (A, μ) and (B, ν). Suppose that (B, ν) is computed by a machine M in polynomial time on ν-average; that is, $\lambda x.\mathrm{Time}_M(x)$ is polynomial on ν-average. The above algorithm N needs at most $O(\mathrm{Time}_f(x) + \mathrm{Time}_M(f(x)))$ steps, where $\mathrm{Time}_f(x)$ denotes the time required to compute $f(x)$. To guarantee the average polynomial-time computability of (A, μ), the values $\hat{\mu}(x)$ and $\hat{\nu}(f(x))$ must be closely related. This last requirement is called a "domination condition" for f.

First recall that the notation $\nu_{f^{-1}}$ denotes the default distribution defined from ν and f by its probability $\hat{\nu}_{f^{-1}}(x) = \hat{\nu}(\{z \mid f(z) = x\})$ for each x.

There are several possibilities for domination conditions. Here is a list of popular conditions:

(i) $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$ [9].

(ii) $\hat{\nu}(z) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for some p-bounded function p.

(iii) $\mu \preceq^{\mathrm{p}} \eta$ and $\hat{\nu} \geq \hat{\eta}_{f^{-1}}$ for some distribution η [36, 109].

By Lemma 3.4.11, condition (ii) is equivalent to the condition that $\mu \preceq^{\mathrm{p}} \eta$ and $\nu \geq \eta_{f^{-1}}$ for some semi-distribution η, and thus, (iii) implies (ii). By the proof of Lemma 3.4.13, (i) implies (ii) if f is p-bounded. Assume that f is p-honest and p-bounded. By Lemma 3.4.12, (ii) implies (iii), and by Lemma 3.4.13, (ii) implies (i).

In most cases, the choice of domination condition is relatively harmless when one proves average **NP**-completeness of a given distributional decision problem, because most known complete problems are isomorphic. In this thesis, however, we shall use the weakest domination condition (ii).

Definition 5.2.1 (Polynomial-Time Many-One Reductions) [60] Given two distributional decision problems (A, μ) and (B, ν), (A, μ) is called *polynomial-time many-one reducible* (p-m-reducible, for short) to (B, ν), denoted by $(A, \mu) \leq_m^{\mathrm{p}} (B, \nu)$, if there exists a function f such that

(i) (Efficiency) $f \in \mathbf{FP}$;

(ii) (Validity) $A = \{x \mid f(x) \in B\}$; and

(iii) (Domination) $\mu \preceq^{\mathrm{p}} \eta$, and $\hat{\nu} \geq \hat{\eta}_{f^{-1}}$ for some semi-distribution η.

The function f is called a *polynomial-time many-one reduction* (p-m-reduction, for short) *function* and is said to reduce (A, μ) to (B, ν).

Gurevich [36] expanded many-one reducibility by allowing a many-one reduction to be polynomial on the average. This reduction is known as *average polynomial-time many-one reducibility.*

Definition 5.2.2 (Average Polynomial-Time Many-One Reductions) [36] Let (A, μ) and (B, ν) be two distributional decision problems. (A, μ) is *average polynomial-time many-one reducible* (avp-m-reducible, for short) to (B, ν), denoted by $(A, \mu) \leq_m^{\mathrm{avp}} (B, \nu)$, if there exists a function f such that

(i) (Efficiency) $(f, \mu) \in \mathrm{Aver}(\mathbf{FP}, *)$;

(ii) (Validity) $A = \{x \mid f(x) \in B\}$; and

(iii) (Domination) $\mu \preceq^{\mathrm{avp}} \eta$, and $\hat{\nu} \geq \hat{\eta}_{f^{-1}}$ for some semi-distribution η.

The condition (iii) on the distributions in the above definition is simply called the *domination condition* for the reduction function f.

Our definition basically follows Gurevich's domination condition. Gurevich [36] defined his many-one reducibility as follows: (A, μ) polynomially many-one reducible to (B, ν) if and only if there exists a **P**-computable function f and a distribution η and a set $C \subseteq \{x \mid \hat{\nu}(x) > 0\}$ such that f reduces $A \cap \{x \mid \hat{\mu}(x) > 0\}$ to B, $\mu \preceq^{\mathrm{p}} \eta$, and $\hat{\nu}\lceil_C(y) = \hat{\eta}_{f^{-1}}(y)$ for all y, where $\nu\lceil_C$ is the distribution satisfying that

$$\hat{\nu}\lceil_C(x) \propto \begin{cases} \hat{\nu}(x) & \text{if } x \in C, \\ 0 & \text{otherwise.} \end{cases}$$

The following lemma is straightforward.

Lemma 5.2.3 *Let D be any set. Given distributions μ and ν, $\mu \preceq^{\mathrm{p}} \nu$ implies $(D, \mu) \leq_m^{\mathrm{p}} (D, \nu)$.*

Proof. Let us consider the identity function f, i.e., $f(x) = x$ for all x. The assumption $\mu \preceq^{\mathrm{p}} \nu$ is equivalent to $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$. Hence, (D, μ) is p-m-reducible to (D, ν) via f. □

The following lemma is a useful tool for proving that the composition of functions is polynomial on μ-average.

Lemma 5.2.4 *Let μ be a distribution and let g be a function on Σ^* such that $\lambda x.|g(x)|$ is polynomial on μ-average. Also let f be a function from Σ^* to $\mathbb{R}^+$ such that f is polynomial on ν-average. If there*

exists a semi-distribution η which avp-dominates μ such that $\hat{\nu}$ majorizes $\hat{\eta}_{g^{-1}}$, then the composition $f \circ g$ is polynomial on μ-average.

Proof. Assume that $\mu \preceq^{\mathrm{avp}} \eta$ and $\hat{\nu} \geq \hat{\mu}_{g^{-1}}$ for some semi-distribution η. Assume that f is p_f on ν-average, and $\lambda x.|g(x)|$ is p_g on μ-average, where p_f and p_g are appropriate increasing polynomials. Since $\mu \preceq^{\mathrm{avp}} \eta$, there exists a function p that is q on μ-average, where q is an increasing polynomial, satisfying $p(x) \cdot \hat{\eta}(x) \geq \hat{\mu}(x)$ for all x.

Fix $r \geq 1$. Let us define

$$s(z) = p_f(p_g(3z) \cdot 6q(3z) \cdot z^2) + c_0,$$

where $c_0 = f \circ g(\lambda)$. Obviously s is a polynomial, and it follows that

$$s(|x| \cdot r) \geq p_f(p_g(|x| \cdot 3r) \cdot 6q(|x| \cdot 3r) \cdot r|x|^2) + c_0.$$

for all strings x. Let $D_r = \{x \in \Sigma^+ \mid p(x) \leq q(|x| \cdot 3r) \wedge |g(x)| \leq p_g(|x| \cdot 3r)\}$. We then have

$$\begin{aligned}\hat{\mu}(\{x \mid f \circ g(x) > s(|x| \cdot r)\}) \quad \leq \quad & \hat{\mu}(\{x \mid p(x) > q(|x| \cdot 3r)\}) + \hat{\mu}(\{x \mid |g(x)| > p_g(|x| \cdot 3r)\}) \\ & + \hat{\mu}(\{x \in D_r \mid \exists z[g(x) = z \wedge f(z) > s(|x| \cdot r)\}).\end{aligned}$$

By our assumption, the first two terms are bounded above by $1/3r$. For simplicity, let T_r represent the third term. It suffices to show that $T_r \leq 1/3r$ in the rest of the proof.

Take an arbitrary $x \in D_r$, say $|x| = n$, and let $z = g(x)$. If $f(g(x)) > s(|x| \cdot r)$, then $f(z) > p_f(p_g(3rn) \cdot 6q(3rn) \cdot rn^2)$. Since $|z| \leq p_g(3rn)$, we have $f(z) > p_f(|z| \cdot 6q(3rn) \cdot rn^2)$. Moreover, it holds that $\hat{\mu}(x) \leq q(3rn) \cdot \hat{\eta}(x)$. Hence, T_r is estimated as follows:

$$\begin{aligned}T_r \quad &\leq \quad \sum_{n=1}^{\infty} \hat{\mu}(\{x \in D_r \cap \Sigma^n \mid \exists z[g(x) = z \wedge f(z) > p_f(|z| \cdot 6q(3rn) \cdot rn^2)]\}) \\ &\leq \quad \sum_{n=1}^{\infty} q(3rn) \cdot \hat{\eta}(\{x \in D_r \cap \Sigma^n \mid \exists z[g(x) = z \wedge f(z) > p_f(|z| \cdot 6q(3rn) \cdot rn^2)]\}) \\ &= \quad \sum_{n=1}^{\infty} q(3rn) \cdot \hat{\eta}_{g^{-1}}(\{z \mid f(z) > p_f(|z| \cdot 6q(3rn) \cdot rn^2)\}).\end{aligned}$$

Since $\hat{\eta}_{g^{-1}}(z) \leq \hat{\nu}(z)$, we conclude:

$$T \leq \sum_{n=1}^{\infty} q(3rn) \cdot \hat{\nu}(\{z \mid f(z) > p_f(|z| \cdot 6q(3rn) \cdot rn^2)\}) \leq \sum_{n=1}^{\infty} \frac{q(3rn)}{6q(3rn) \cdot rn^2} = \frac{\pi^2}{36r} < \frac{1}{3r}.$$

□

In the following, we demonstrate some of the basic properties of many-one reducibilities.

Proposition 5.2.5 *Let (A, μ), (B, ν), and (A_i, μ_i), $i = 1, 2, 3$, be distributional decision problems.*

1. *The $\leq_m^{\mathrm{p}}$ and $\leq_m^{\mathrm{avp}}$ are reflexive; i.e., $\alpha \in \{\mathrm{p}, \mathrm{avp}\}$, $(A, \mu) \leq_m^{\alpha} (A, \mu)$.*

2. The $\leq_m^{\mathrm{p}}$ *implies* $\leq_m^{\mathrm{avp}}$*; i.e.,* $(A,\mu) \leq_m^{\mathrm{p}} (B,\nu)$ *implies* $(A,\mu) \leq_m^{\mathrm{avp}} (B,\nu)$*.*

3. The $\leq_m^{\mathrm{p}}$ *and* $\leq_m^{\mathrm{avp}}$ *are transitive; i.e., for* $\alpha \in \{\mathrm{p}, \mathrm{avp}\}$*, if* $(A_1,\mu_1) \leq_m^{\alpha} (A_2,\mu_2)$ *and* $(A_2,\mu_2) \leq_m^{\alpha} (A_3,\mu_3)$*, then* $(A_1,\mu_1) \leq_m^{\alpha} (A_3,\mu_3)$*.*

Proof. (1)-(2) Clear from the definitions.

(3) We shall show the transitivity of $\leq_m^{\mathrm{avp}}$. First we assume that $(A_1,\mu_1) \leq_m^{\mathrm{p}} (A_2,\mu_2)$ via f_1 and $(A_2,\mu_2) \leq_m^{\mathrm{avp}} (A_3,\mu_3)$ via f_2. The domination conditions for f_1 and f_2 ensure the existence of two semi-distributions η_1 and η_2 such that $\mu_1 \preceq^{\mathrm{avp}} \eta_1$, $\mu_2 \preceq^{\mathrm{avp}} \eta_2$, $\hat{\mu}_2 \geq \lambda z.\hat{\eta}_1(f_1^{-1}(z))$, and $\hat{\mu}_3 \geq \lambda z.\hat{\eta}_2(f_2^{-1}(z))$.

Since f_1 and f_2 are many-one reductions, we define f as $f(x) = f_2 \circ f_1(x)$. Then function f reduces A_1 to A_3. In the rest of the proof, we shall check the domination condition for f.

From $\mu_1 \preceq^{\mathrm{p}} \eta_1$, it follows that there exists a function p_1, which is polynomial on μ_1-average, satisfying that $p_1(x) \cdot \hat{\eta}_1(x) \geq \hat{\mu}_1(x)$ for all x. Similarly, there is another function p_2 that is polynomial on μ_2-average such that $p_2(x) \cdot \hat{\eta}_2(x) \geq \hat{\mu}_2(x)$ for all x. We assume without loss of generality that $p_1(x) \geq 1$ and $p_2(x) \geq 1$ for all x.

Let us define $p(x) = p_2(f_1(x)) \cdot p_1(x)$ for all x. Notice that p is polynomial on μ_1-average. To see this, we first note that $p_2 \circ f_1$ is polynomial on μ-average by Lemma 5.2.4 because $\lambda x.|f_1(x)|$ is polynomial on μ_1-average and p_2 is polynomial on μ_2-average. Since p_1 is also polynomial on μ_1-average, p turns out to be polynomial on μ_1-average. Set $\hat{\eta}(x) = \hat{\mu}_1(x)/p(x)$ for all x. Clearly η is a semi-distribution.

For any string y in $\mathrm{ran}(f)$, letting $D_y = f_2^{-1}(y) \cap \mathrm{ran}(f_1)$, it follows that

$$\hat{\mu}_3(y) \geq \hat{\eta}_2(f_2^{-1}(y)) \geq \hat{\eta}_2(f_2^{-1}(y) \cap \mathrm{ran}(f_1)) \geq \sum_{w \in D_y} \hat{\eta}_2(w).$$

For each string w in $\mathrm{ran}(f_1)$,

$$\hat{\eta}_2(w) \geq \frac{\hat{\mu}_2(w)}{p_2(w)} \geq \frac{\hat{\eta}_1(f_1^{-1}(w))}{p_2(w)} \geq \sum_{x \in f_1^{-1}(w)} \frac{\hat{\mu}_1(x)}{p_1(x) \cdot p_2(f_1(x))} = \sum_{x \in f_1^{-1}(w)} \frac{\hat{\mu}_1(x)}{p(x)} = \hat{\eta}(f_1^{-1}(w)),$$

and thus, we obtain that $\hat{\eta}(w) \geq \hat{\eta}(f_1^{-1}(w))$. Hence,

$$\hat{\mu}_3(y) \geq \sum_{w \in D_y} \hat{\eta}(f_1^{-1}(w)) = \hat{\eta}\left(\bigcup_{w \in D_y} f_1^{-1}(w)\right) = \hat{\eta}(f^{-1}(y)).$$

□

Theorem 5.2.6 *Let* $\mathcal{C} \in \{\mathbf{P}, \mathbf{NP}, \mathbf{RP}, \mathbf{BPP}, \mathbf{PSPACE}\}$*. The average complexity class* $\mathrm{Aver}(\mathcal{C},*)$ *is closed downward under* $\leq_m^{\mathrm{avp}}$*-reductions.*

Proof. We begin with the closure property of $\mathrm{Aver}(\mathbf{P},*)$ under $\leq_m^{\mathrm{avp}}$-reductions. Let us assume that $(A,\mu) \leq_m^{\mathrm{avp}} (B,\nu)$ via f and $(B,\nu) \in \mathrm{Aver}(\mathbf{P},*)$. Note that f is computable in time polynomial on μ-average. Let M be a deterministic Turing machine which computes B in polynomial time on ν-average.

We can construct another deterministic Turing machine N which, on input x, computes $f(x)$ first and then simulates M on input $f(x)$. This machine N actually computes A, and its running time on input x needs at most

$$c \cdot (\mathrm{Time}_f(x) + \mathrm{Time}_M(f(x)) + 1),$$

where $\mathrm{Time}_f(x)$ denotes the time required to compute the value $f(x)$, and c denotes some appropriate constant independent of x.

Notice that $\lambda x.|f(x)|$ is polynomial on μ-average. Since $\lambda x.\mathrm{Time}_M(x)$ is polynomial on ν-average, the domination condition for f implies, by Lemma 5.2.4, that $\lambda x.\mathrm{Time}_M(f(x))$ is also polynomial on μ-average. Therefore, N is polynomial-time bounded on μ-average.

The proofs for the classes $\mathcal{C} \in \{\mathbf{NP}, \mathbf{RP}, \mathbf{BPP}, \mathbf{PSPACE}\}$ are similar. □

5.2.2 Polynomial Time Isomorphism

A *polynomial-time Isomorphism* in worst-case complexity theory is a **P**-computable, p-invertible bijection which p-m-reduces a set A to another set B. If such an isomorphism exists, then we say that A is *p-isomorphic to* B. Interestingly, most **NP**-complete sets are known to be p-isomorphic.

Wang and Belanger [112] introduced a polynomial-time isomorphism between two distributional problems. This subsection will introduce the notion of *polynomial-time isomorphism*.

We first introduce a notion of one-one reducibility.

Definition 5.2.7 (Polynomial-Time One-One Reductions) For two distributional decision problems (A, μ) and (B, ν), the problem (A, μ) is *polynomial-time one-one reducible* (p-1-reducible, for short) to (B, ν), denoted by $(A, \mu) \leq_1^{\mathrm{p}} (B, \nu)$, if there exists a one-one, **P**-computable reduction f which p-m-reduces (A, μ) to (B, ν).

Since the reduction f is one-one, the domination condition for f is simply expressed as $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$, or equivalently $\mu \preceq^{\mathrm{p}} \nu \circ f$ by Lemma 3.4.10(2).

We then introduce a polynomial-time isomorphism among distributional decision problems.

Definition 5.2.8 (Polynomially Isomorphic) [112] For two distributional decision problems (A, μ) and (B, ν), (A, μ) is *polynomially isomorphic* (p-isomorphic, for short) to (B, ν) if there exists a **P**-computable, p-invertible bijection f on Σ^* such that $(A, \mu) \leq_1^{\mathrm{p}} (B, \nu)$ via f and $(B, \nu) \leq_1^{\mathrm{p}} (A, \mu)$ via f^{-1}. This f is called a *polynomial-time isomorphism* (p-isomorphism, for short).

Berman and Hartmanis [10] show by an analogy of Myhill's isomorphism theorem in recursion theory that, for two functions which are one-one, length-increasing, **P**-computable, and p-invertible, if $A \leq_1^{\mathrm{p}} B$ via f and $B \leq_1^{\mathrm{p}} A$ via g, then A and B are p-isomorphic. In fact, however, we need only the condition that $f \circ g$ and $g \circ f$ are length-increasing instead of both f and g being length-increasing. Wang and Belanger [112]

give an analogous result in average-case setting.

Proposition 5.2.9 **[112]** *Let (A,μ) and (B,ν) be two distributional decision problems. Let f,g be one-one, $\mathbf{P}$-computable, and p-invertible, and assume that $f\circ g$ and $g\circ f$ are length-increasing. Moreover, assume that $\mu \approx^{\mathrm{p}} \nu\circ f$ and $\nu \approx^{\mathrm{p}} \mu\circ g$. If $(A,\mu) \leq_1^{\mathrm{p}} (B,\nu)$ via f and $(B,\nu) \leq_1^{\mathrm{p}} (A,\mu)$ via g, then (A,μ) and (B,ν) are p-isomorphic.*

Proof. Assume that f, g, μ, and ν satisfy the conditions of the proposition. Let p be a polynomial which is a time-bound of $\mathbf{P}$-computable functions, f, g, f^{-1}, and g^{-1}. We first define two sets R_1 and R_2 as follows:

$$R_1 = \{(g\circ f)^k(x) \mid k \geq 0, x \notin g(\Sigma^*)\}; \text{ and}$$

$$R_2 = \{g\circ(f\circ g)^k(x) \mid k \geq 0, x \notin f(\Sigma^*)\},$$

where $(g\circ f)^k(x)$ means k applications of the function $g\circ f$ to x, and in particular, $(g\circ f)^0(x) = x$.

We claim that $R_1 \cup R_2 = \Sigma^*$ and $R_1 \cap R_2 = \emptyset$.

Claim 7 *$R_1 \cap R_2 = \emptyset$.*

Proof of Claim. Assume that there is an element x in $R_1 \cap R_2$. We take two strings y and z such that $x = (g\circ f)^k(y) = g\circ(f\circ g)^m(z)$, $y \notin g(\Sigma^*)$, and $z \notin f(\Sigma^*)$ for some $k,m \geq 0$. Notice that $x \in g(\Sigma^*)$. Obviously $k > 0$, since otherwise, $y = x$ and $x \notin g(\Sigma^*)$, a contradiction. Hence, we have $k > 0$. In the case where $m \geq k$, we have $x = g\circ(f\circ g)^m(z) = (g\circ f)^m \circ g(z)$. As $x = (g\circ f)^m(y)$ and $g\circ f$ is one-one, it follows that $y = (g\circ f)^{m-k}\circ g(z) = g\circ(f\circ g)^{m-k}(z)$. Clearly the last term belongs to $g(\Sigma^*)$, so y is in $g(\Sigma^*)$, a contradiction. Hence, $m < k$. Using $g\circ(f\circ g)^m(z) = x$, we have $x = (g\circ f)^k(y) = g\circ(f\circ g)^{k-1}\circ f(y)$, and thus $(f\circ g)^m(z) = (f\circ g)^{k-1}(f(y))$. Thus, $z = (f\circ g)^{k-m-1}(f(y)) = f\circ(g\circ f)^{k-m-1}(y)$. This implies that $z \in f(\Sigma^m)$, a contradiction. Therefore, $R_1 \cap R_2 = \emptyset$. ∎

Claim 8 *$R_1 \cup R_2 = \Sigma^*$.*

Proof of Claim. Take an arbitrary x in Σ^*. Assume that $x = (g\circ f)^k(y)$ and $x = g\circ(f\circ g)^m(z)$ for some y, z, m, and k. We also assume that k,m are maximal. If either $y \notin g(\Sigma^*)$ or $z \notin f(\Sigma^*)$, then x is in $R_1 \cup R_2$. Now we assume to the contrary that $y = g(w_y)$ and $z = f(w_z)$ for some appropriate w_y and w_z. From $y = g(w_y)$, it follows that $(g\circ f)^k(y) = g\circ(f\circ g)^k(w_y)$. From $z = f(w_z)$, it follows that $g\circ(f\circ g)^m(z) = (g\circ f)^{m+1}(w_z)$. Both imply the same x, and thus we have $(g\circ f)^k(y) = (g\circ f)^{m+1}(w_z)$ and $g\circ(f\circ g)^k(w_y) = g\circ(f\circ g)^m(z)$. By the maximality of k and m, we conclude both that $k = m+1$ and $k = m$. This is a contradiction. ∎

It is possible to check whether $z \in R_1$ or $z \in R_2$ within $O((|z|+2)p(|z|))$ steps. This is shown as

follows. Notice that $f^{-1} \circ g^{-1}$ is **P**-computable because $\mathrm{ran}(f)$ and $\mathrm{ran}(f)$ are both **P**-computable, and, more important, it is length-decreasing (i.e., $|z| > |f^{-1} \circ g^{-1}|$) because $g \circ f$ is length-increasing. Let us consider the strictly descending chain

$$|z| > |f^{-1} \circ g^{-1}(z)| > |(f^{-1} \circ g^{-1})^2(z)| > \cdots$$

until $f^{-1} \circ g^{-1}$ can no longer be applied. This chain consists of at most $|z| + 1$ elements. Let $\tilde{z}$ be the last element of this descending chain. Hence, $z = (g \circ f)^k(\tilde{z})$ for some $k \geq 0$. Notice that if $\tilde{z} \in R_1$, then all other elements are in R_1 by the definition of R_1. In particular, $z \in R_1$ if and only if $z' \in R_1$. To check whether $z \in R_1$, we would check whether g^{-1} cannot be applied to z, and thus we need $O((|z| + 2)p(|z|))$ steps. Similarly, we can determine whether $z \in R_2$ in $O((|z| + 2)p(|z|))$ steps. As a consequence, R_1 and R_2 are both **P**-computable.

Similarly, we define S_1 and S_2 as follows:

$$S_1 = \{(f \circ g)^k(x) \mid k \geq 0, x \notin f(\Sigma^*)\}; \text{ and}$$

$$S_2 = \{f \circ (g \circ f)^k(x) \mid k \geq 0, x \notin g(\Sigma^*)\}.$$

We also have $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = \Sigma^*$. Moreover, S_1 and S_2 are **P**-computable.

Let us define the desired p-isomorphism h as follows:

$$h(z) = \begin{cases} f(z) & \text{if } z \in R_1, \\ g^{-1}(z) & \text{if } z \in R_2. \end{cases}$$

This function h is total because $R_1 \cup R_2 = \Sigma^*$. Also h is one-one because f and g^{-1} are one-one, and h is also onto because $h(R_1) = S_2$ and $h(R_2) = S_1$. It is not hard to show that

$$h^{-1}(z) = \begin{cases} f^{-1}(z) & \text{if } z \in S_2, \\ g(z) & \text{if } z \in S_1. \end{cases}$$

Hence, h is total and thus a bijection on Σ^*.

We can easily see that h reduces A to B because $h(x) = f(x)$ for all $x \in R_1$, and $h(x) = g^{-1}(x)$ for all $x \in R_2$. Similarly, h^{-1} reduces B to A.

What remains is to check the domination conditions for h and h^{-1}. It is sufficient to prove that $\mu \preceq^{\mathrm{p}} \nu \circ h$ and $\nu \preceq^{\mathrm{p}} \mu \circ h^{-1}$. We shall show that $\mu \preceq^{\mathrm{p}} \nu \circ h$. Since $\mu \preceq^{\mathrm{p}} \nu \circ f$, let s_0 be a polynomial such that $\hat{\mu}(z) \leq s_0(|z|) \cdot \hat{\nu}(f(z))$ for all z. Similarly, since $\mu \circ g \preceq^{\mathrm{p}} \nu$, there is a polynomial s_1 such that $\hat{\mu}(g(z)) \leq s_1(|z|) \cdot \hat{\nu}(z)$ for all z. Recall that g is p-honest, and thus we can take a polynomial q such that $|z| \leq q(|g(z)|)$ for all z. Let $s(n) = s_0(n) + s_1(q(n))$. If $z \in R_1$, then $\hat{\mu}(z) \leq s_0(|z|) \cdot \hat{\mu}(f(z)) \leq s(|z|) \cdot \hat{\nu}(h(z))$. If $z \in R_2$, then let $z = g(w)$, so we have $\hat{\mu}(g(w)) \leq s_0(|w|) \cdot \hat{\nu}(w) \leq s_0(q(|g(w)|)) \cdot \hat{\nu}(w)$. This implies that $\hat{\nu}(z) \leq s(|z|) \cdot \hat{\nu}(h(z))$. Therefore, $\mu \preceq^{\mathrm{p}} \circ h$. Similarly, we can show that $\nu \preceq^{\mathrm{p}} \mu \circ h^{-1}$.

This completes the proof. □

The reader who is interested in other types of many-one reducibilities may refer to [36, 9, 30, 51, 44, 32, 108].

5.2.3 Deterministic Turing Reducibility

We turn our attention to Turing reducibility. In worst-case complexity theory, Cook [22] first formulated deterministic polynomial-time Turing reductions to show that SAT, the satisfiability problem, is one of the hardest problems that is in **NP**. A Turing reduction from a set A to another set B is a deterministic oracle Turing machine that computes set A with the help of set B as an oracle.

In average-case complexity theory, Ben-David, Chor, Goldreich, and Luby [9] introduced a similar notion of deterministic Turing reducibility among distributional decision problems. We begin with this deterministic Turing reducibility. Let us recall that the notation $Q(M, B, x)$ denotes the set of strings queried by an oracle Turing machine M with oracle B on input x.

Definition 5.2.10 (Deterministic Turing Reductions) [9] Let (A, μ) and (B, ν) be distributional decision problems.

1. (A, μ) is *polynomial-time Turing reducible* (p-T-reducible, for short) to (B, ν), denoted by $(A, \mu) \leq_T^{\mathrm{p}} (B, \nu)$, if there exist a deterministic oracle Turing machine M and a semi-distribution η such that
   (i) (Efficiency) M with oracle B is polynomial-time bounded;
   (ii) (Validity) $A = L(M, B)$; and
   (iii) (Domination) $\mu \preceq^{\mathrm{p}} \eta$ and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{x \mid z \in Q(M, B, x)\})$.
2. (A, μ) is *average polynomial-time Turing reducible* (avp-T-reducible, for short) to (B, ν), symbolically $(A, \mu) \leq_T^{\mathrm{avp}} (B, \nu)$, if there exist a deterministic oracle Turing machine M and a semi-distribution η such that
   (i) (Efficiency) M with oracle B is polynomial-time bounded on μ-average;
   (ii) (Validity) $A = L(M, B)$; and
   (iii) (Domination) $\mu \preceq^{\mathrm{avp}} \eta$ and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{x \mid z \in Q(M, B, x)\})$.

   The condition (iii) is called the *domination condition for the reduction* M.
   Originally Ben-David *et al.* [9] used a stronger domination condition:

$$\hat{\mu}(\{x \mid z \in Q(M, B, x)\}) \leq \hat{\nu}(z) \cdot p(z)$$

for some polynomial p.

Proposition 5.2.11 *1. The relation $\leq_m^{\mathrm{p}}$ implies $\leq_T^{\mathrm{p}}$, and $\leq_m^{\mathrm{avp}}$ implies $\leq_T^{\mathrm{avp}}$.*

2. The relation $\leq_T^{\mathrm{p}}$ and $\leq_T^{\mathrm{avp}}$ are reflexive.

3. The relation $\leq_T^{\mathrm{p}}$ implies $\leq_T^{\mathrm{avp}}$.

4. The relation $\leq_T^{\mathrm{p}}$ and $\leq_T^{\mathrm{avp}}$ are transitive.

Proof. (1)-(3) Clear from the definitions.

(4) Here we show that $\leq_T^{\rm avp}$ is transitive. The proofs for the transitivity of the other reducibilities are analogous. Now we assume that (D_1, μ_1) is avp-T-reducible to (D_2, μ_2) via a reduction M_1 and a semi-distribution ν_1, and also assume that (D_2, μ_2) is avp-T-reducible to (D_3, μ_3) via a reduction M_2 and a semi-distribution ν_2. We assume that $Q(M_1, D_2, x) \neq \emptyset$ for infinitely many x since, otherwise, $(D_1, \mu_1) \in {\rm Aver}(\mathbf{P}, *)$ and, thus trivially (D_1, μ_1) is avp-T-reducible to (D_3, μ_3). In what follows, we shall show that $(D_1, \mu_1) \leq_T^{\rm avp} (D_3, \mu_3)$.

By definition, there exist two functions f_1 and f_2 which are polynomial on μ_1-average and on μ_2-average, respectively, such that $f_1(x) \cdot \hat{\nu}_1(x) \geq \hat{\mu}_1(x)$ and $f_2(x) \cdot \hat{\nu}_2(x) \geq \hat{\mu}_2(x)$ for all x. Without loss of generality, we assume that $f_1(x) \geq 2$ and $f_2(x) \geq 2$ for all strings x.

We define a new machine M as follows: on input x, M deterministically simulates M_1 on x, and whenever M_1 queries a string y, M deterministically simulates M_2 on input y. In the case where x is the empty string λ, M must be designed not to query any strings (if M_2 queries some strings to oracle D_3, then their oracle answers are encoded into a program of M). Clearly $D_1 = L(M, D_3))$. Note that

$$\mathrm{Time}_M^{D_3}(x) \leq c \cdot \left(\mathrm{Time}_{M_1}^{D_2}(x) + \sum_{y \in Q(M_1, D_2, x)} \mathrm{Time}_{M_2}^{D_3}(y) + 1 \right)$$

for some constant $c > 0$.

Let $f(x) = f_1(x) \cdot \max_{y \in Q(M_1, D_2, x)} f_2(y)$. Obviously, for each x, $f(x) \geq 1$, and thus $\hat{\mu}_1(x) < f(x)$. Now we choose a distribution ν:

$$\hat{\nu}(x) = \begin{cases} \frac{\hat{\mu}_1(x)}{f(x)} & \text{if } x \neq \lambda, \\ 1 - \sum_{y : y \neq \lambda} \hat{\nu}(y) & \text{if } x = \lambda. \end{cases}$$

In particular, $\hat{\nu}(\lambda) > 0$. Let $c_0 = \frac{\hat{\mu}(\lambda)}{\hat{\nu}(\lambda)}$. Then, we have $(f(x) + c_0)\hat{\nu}(x) \geq \hat{\mu}_1(x)$ for all $x \neq \lambda$. For each pair (x, y), it follows that $z \in Q(M_2, D_3, y)$ and $y \in Q(M_1, D_2, x)$ if and only if $z \in Q(M, D_3, x)$ and $y \in Q(M_1, D_2, x)$. Note also that

$$\hat{\mu}_2(y) \geq \hat{\nu}_1(\{x \mid y \in Q(M_1, D_2, x)\}) \geq \sum_{x : y \in Q(M_1, D_2, x)} \frac{\hat{\mu}_1(x)}{f_1(x)}.$$

Then, for all $z \in \bigcup_x Q(M, D_3, x)$,

$$\begin{aligned} \hat{\mu}_3(z) &\geq \hat{\nu}_2(\{y \mid z \in Q(M_2, D_3, y)\}) \geq \sum_{y : z \in Q(M_2, D_3, y)} \frac{\hat{\mu}_2(y)}{f_2(y)} \\ &\geq \sum_{y : z \in Q(M_2, D_3, y)} \frac{1}{f_2(y)} \sum_{x : y \in Q(M_1, D_2, x)} \frac{\hat{\mu}_1(x)}{f_1(x)} = \sum_{x : z \in Q(M, D_3, x)} \sum_{y \in Q(M_1, D_2, x)} \frac{1}{f_2(y)} \cdot \frac{\hat{\mu}_1(x)}{f_1(x)} \\ &\geq \sum_{x : z \in Q(M, D_3, x)} \frac{\hat{\mu}_1(x)}{f_1(x) \cdot \max_{y \in Q(M_1, D_2, x)} f_2(y)} = \sum_{x : z \in Q(M, D_3, x)} \frac{\hat{\mu}_1(x)}{f(x)} \\ &= \hat{\nu}(\{x \mid z \in Q(M, D_3, x)\}). \end{aligned}$$

Next we shall show that M and f are both polynomial on μ_1-average. We first see that M is polynomial-time bounded on μ_1-average. Let $h(x) = \sum_{y \in Q(M_1, D_2, x)} \mathrm{Time}_{M_2}^{D_3}(y)$ for each x. To complete the proof, by

Lemmas 3.3.13 and 3.3.12, it is sufficient to show that h is polynomial on μ_1-average since $\lambda x.\mathrm{Time}_{M_1}^{D_2}(x)$ is polynomial on μ_1-average; however, this is not difficult to see.

The proof that f is polynomial on μ_1-average is similar, and thus the claim is established. □

The transitivity of avp-T-reducibility implies the closure property of $\mathrm{Aver}(\mathbf{P}, *)$ under avp-T-reductions.

Theorem 5.2.12 [9] *The class* $\mathrm{Aver}(\mathbf{P}, *)$ *is closed downward under avp-T-reductions.*

Proof. Assume that $(A, \mu) \leq_T^{\mathrm{avp}} (B, \nu)$ for some $(B, \nu) \in \mathrm{Aver}(\mathbf{P}, *)$. Note that

$$\mathrm{Aver}(\mathbf{P}, *) = \{(A, \mu) \mid (A, \mu) \leq_T^{\mathrm{avp}} (\emptyset, \nu_{\mathrm{stand}})\}.$$

Since $(B, \nu) \leq_T^{\mathrm{avp}} (\emptyset, \nu_{\mathrm{stand}})$, by Proposition 5.2.11(4), we obtain $(A, \mu) \leq_T^{\mathrm{avp}} (\emptyset, \nu_{\mathrm{stand}})$. Thus, $(A, \mu) \in \mathrm{Aver}(\mathbf{P}, \mathcal{F})$. □

5.3 Many-One Complete Problems

We have introduced two different types of reducibilities in the previous section which play a significant role in measuring the relative complexity of any two distributional decision problems. If a distributional problem (A, μ) is reducible to another distributional problem (B, ν), then (B, ν) is considered at least as hard as (A, μ) to solve. We wish to see the hardest problems in $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ in this sense.

The notion of *complete problems* was introduced into computational complexity theory in the early 1970's, and subsequently many problems have been found to be complete for **NP**. We generalize the definition of "completeness" to our setting below.

Definition 5.3.1 (Complete Problems) Let $\leq_\alpha$ be any reducibility and let $\mathcal{C}$ be a class of distributional decision problems.

1. A distributional problem (D, μ) is $\leq_\alpha$-*hard* for $\mathcal{C}$ if every problem (E, η) in $\mathcal{C}$ is $\leq_\alpha$-reducible to (D, μ).

2. A distributional problem (D, μ) is $\leq_\alpha$-*complete* for $\mathcal{C}$ if it is in $\mathcal{C}$ and is $\leq_\alpha$-hard for $\mathcal{C}$.

This section will show that several important distributional decision problems are p-m-complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$, and hence that they are among the hardest problems to solve.

Intriguingly, Belanger and Wang [6] pointed out that most known p-m-complete problems for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ are actually p-isomorphic. They proposed an average-case version of a conjecture given by Berman and Hartmanis [10], known as (Berman-Hartmanis) isomorphism conjecture, that asserts that all p-m-complete distributional problems for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ are p-isomorphic. We refer to this conjecture as the *Isomorphism Conjecture*.

5.3.1 Randomized Bounded Halting problem

One of the most useful distributional decision problems is the *randomized bounded halting problem*. To describe it, we assume an effective enumeration of all nondeterministic Turing machines, say $\{M_i\}_{i\in\mathbb{N}}$. The *randomized bounded halting problem* (RBHP) is the distributional problem $(\mathrm{BHP}, \mu_{\mathrm{BHP}})$ that is based on the bounded halting problem

$$\mathrm{BHP} = \{\langle s_i, x, 1^n\rangle \mid M_i \text{ accepts } x \text{ in less than } n \text{ steps }\},$$

where s_i is the string corresponding to i and based on the default probability

$$\hat{\mu}_{\mathrm{BHP}}(s_i, x, 1^n) = \hat{\nu}_{\mathrm{stand}}(s_i) \cdot \hat{\nu}_{\mathrm{stand}}(x) \cdot \hat{\nu}_{\mathrm{tally}}(1^n).$$

Intuitively, we independently pick up a string s_i, a string x, and a unary string 1^n at random. Clearly μ_{BHP} is supportive and **P**-computable. We remark here that the choice of $\hat{\nu}_{\mathrm{tally}}(1^n)$ is essential.

In the following theorem, we shall prove that RBHP is $\leq^{\mathrm{p}}_m$-complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. This theorem has been proven in numerous ways (see e.g., [9, 30, 36, 111]). The proof below follows the argument given by Wang and Belanger [111].

Theorem 5.3.2 **[36]** RBHP *is* $\leq^{\mathrm{p}}_m$*-complete for* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$.

Proof. We first note that RBHP is in $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ because BHP belongs to **NP** and μ_{BHP} is **P**-computable.

Consider any distributional decision problem (D, μ) from $\mathrm{Aver}(\mathbf{NP}, \mathbf{P}\text{-comp})$. There exists a polynomial-time nondeterministic Turing machine M which accepts D. For every set $D \in \mathbf{NP}$ and every distribution $\mu \in \mathbf{P}$-comp, we next show that $(D, \mu) \leq^{\mathrm{p}}_m (\mathrm{BHP}^k, \mu_{\mathrm{BHP}})$. Let g be the one-one, p-invertible, **P**-computable function of Lemma 4.2.7(2). Note that $|g(x)| \leq q(|x|)$ for some absolute polynomial q, and that $\hat{\mu}(x) < 2^{-|g(x)|+2}$ for all x. Now let us consider the machine N that simulates M in the following fashion:

begin nondeterministic Turing machine N
  input y
  compute a string x such that $y = g(x)$ by binary search
    (this is done in time polynomial in $|x|$)
  if such an x exists **then** nondeterministically simulate M on input x
    else reject
end.

Since N is a polynomial-time nondeterministic Turing machine, we take an index i such that $L(M_i) = L(N)$. Let p be a polynomial time-bound of M_i. The desired reduction f is now defined as $f(x) = \langle s_i, g(x), 1^{p(|x|)}\rangle$. It is not difficult to see that f is one-one and reduces D to BHP. It is sufficient to check that f satisfies the domination condition. Let s be any polynomial such that $\hat{\nu}_{\mathrm{stand}}(s_i) \cdot s(n) \geq 256(q(n)+1)^2(p(n)+1)^2$ for all $n \in \mathbb{N}$. Note that i is a constant and does not depend on n. Then, we have:

$$s(|x|) \cdot \hat{\mu}_{\mathrm{BHP}}(f(x)) \quad = \quad s(|x|) \cdot \hat{\mu}_{\mathrm{BHP}}(\langle s_i, g(x), 1^{p(|x|)}\rangle)$$

$$\begin{aligned} &= s(|x|) \cdot \hat{\nu}_{\mathrm{stand}}(s_i) \cdot \hat{\nu}_{\mathrm{stand}}(g(x)) \cdot 2^{-2\lfloor \log(p(|x|)) \rfloor - 1} \\ &\geq \frac{s(|x|) \cdot \hat{\nu}_{\mathrm{stand}}(s_i)}{256(p(|x|)+1)^2(|g(x)|+1)^2} \cdot 2^{-|g(x)|+2} \\ &\geq 2^{-|g(x)|+2} \geq \hat{\mu}(x). \end{aligned}$$

Hence, $(D, \mu) \leq^{\mathrm{p}}_m (\mathrm{BHP}, \mu_{\mathrm{BHP}})$. □

In Section 6.5, we shall generalize the problem RBHP in order to show that the generalized RBHP is also complete for $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathbf{P}\text{-comp})$ for each $k > 0$.

5.3.2 Randomized Bounded Tiling problem

The *randomized bounded tiling problem* (RBTP) is the first problem discovered by Levin [60] to be $\leq^{\mathrm{p}}_m$-complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. RBTP is the distributional problem $(\mathrm{BTP}, \mu_{\mathrm{BTP}})$ that is defined as follows. A *tile* is a quadruple $[u, v, x, w]$ of strings, where u is called "left," v "top," x "right," and w "bottom." We use the notation $left[u, v, x, w]$ to denote the left element u. Similar notations are used for "top," "right," and "bottom." Let S_n be the $n \times n$ square $\{1, \ldots, n\} \times \{1, \ldots, n\}$. Let T be a set of tiles. A function f from S_n to T is called a *T-tiling* of S_n if $left[f(i+1, j)] = right[f(i, j)]$ and $bottom[f(i, j+1)] = top[f(i, j)]$ for all i, j with $1 \leq i < n$ and $1 \leq j < n$. A sequence $\langle t_1, \ldots, t_k \rangle$ is a *T-row of length k* if $t_i \in T$ for all i with $1 \leq i \leq n$ and $left[s_{j+1}] = right[s_j]$ for all j with $1 \leq j < n$. Let

$$\begin{aligned} \mathrm{BTP} = \{\langle T, 1^n, 1^k, \langle t_1, \ldots, t_k \rangle \rangle \mid\ & \langle t_1, \ldots, t_k \rangle \text{ is a } T\text{-row of length } k,\ 1 \leq k \leq n, \\ & \exists f[f \text{ is a } T\text{-tiling of } S_n \text{ and } \forall i[1 \leq i \leq k \rightarrow f(1,1) = t_i]]\}. \end{aligned}$$

Fix a positive $\mathbf{P}$-computable distribution ν for a set T of tiles and let

$$\hat{\mu}_{\mathrm{BTP}}(T, 1^n, 1^k, \langle t_1, \ldots, t_k \rangle) = \begin{cases} \hat{\nu}(T) \cdot \hat{\nu}_{\mathrm{tally}}(1^n) \cdot \frac{1}{n} \cdot \prod_{i=1}^{k} \frac{1}{\|T_i\|} & \text{if } 1 \leq k \leq n \text{ and} \\ & T_i \neq \emptyset \text{ for all } i \text{ with } 1 \leq i \leq k, \\ 0 & \text{otherwise,} \end{cases}$$

where $T_i = \{t \in T \mid left[t] = right[t_i]\}$. We remark here that the choice of a default distribution ν for tiles is not important because it does not affect the proof of Theorem 5.3.3.

It is shown that RBTP is p-m-reducible to RBHP [60, 36].

Theorem 5.3.3 **[112]** RBTP *is p-isomorphic to* RBHP.

Proof. It is known that BTP is **NP**-complete. Thus, RBTP is in $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. In the following, we shall construct two one-one, length-increasing, p-invertible, **P**-computable reductions f and g from RBHP to RBTP and from RBTP to RBHP, respectively, with the condition that $\mu_{\mathrm{BHP}} \approx^{\mathrm{p}} \mu_{\mathrm{BTP}} \circ f$ and $\mu_{\mathrm{BTP}} \approx^{\mathrm{p}} \mu_{\mathrm{BHP}} \circ g$. This is sufficient to show the theorem, because Proposition 5.2.9 yields the existence of a p-isomorphism between RBHP and RBTP.

Since $\mathrm{BHP} \in \mathbf{NP}$, there is a nondeterministic Turing machine M accepting BHP in polynomial time. From the Distribution Controlling Lemma (Lemma 4.2.7), there exists a total, one-one, p-invertible, **P**-

computable function h such that $4 \cdot 2^{-|h(x)|} \leq \hat{\mu}_{\mathrm{BHP}}(x) \leq 20 \cdot 2^{-|h(x)|}$ for all input x. Take a polynomial q such that $|h(x)| \leq q(|x|)$ for all x.

For each string w, let $w_L = 0^m 1 s_{|w|} w$, where $m = |s_{|w|}|$. Notice that if a string z is given, then we can uniquely determine w_L such that $w_L \sqsubseteq z$ if one exists. Let M' be the following algorithm:

begin algorithm M'
  input z
  if there is no w such that $w_L \sqsubseteq z$ **then** reject
    (Assume that there is the unique w such that $w_L \sqsubseteq z$.)
  check if $w = h(x)$ for some x by a binary search
    (This is done in time polynomial in $|x|$.)
  if no such x exists **then** reject
  simulate M on input x
end.

Now write $\tilde{x}$ instead of $h(x)$ for brevity. We then have $\tilde{x}_L = 0^m 1 s_{|h(x)|} h(x)$, where $m = |s_{|h(x)|}|$. Note that

$$|\tilde{x}_L| = |0^m 1 s_{|h(x)|} h(x)| = 2|s_{|h(x)|}| + |h(x)| + 1 = 2|\log(|h(x)|) + |h(x)| + 1 \leq 2|\log(q(|x|)) + |h(x)| + 1,$$

where $m = |s_{|h(x)|}|$. Moreover, it is clear that $x \in L(M)$ if and only if $\tilde{x}_L z \in L(M')$ for any string z.

Recall the (formal) definition of Turing machines given in Section 2.3. Let $\langle Q, q_0, acc, rej, \delta, B \rangle$ define the machine M, where Q is the set of states, q_0 the initial state, acc the accepting state, rej the rejecting state, δ the transition function, and B the blank symbol. Let M be any nondeterministic Turing machine which runs in polynomial time. Let $T_{M'}$ contain the following tiles (T1)-(T5):

(T1) $[\$, \langle q_0, a \rangle, \#, \$]$, where q_0 is the initial state of M' and $a \in \{0, 1\}$.

(T2) $[\#, a, \#, \$]$, where $a \in \{0, 1\}$.

(T3) $[\%, c, \%, c]$, $[\%, c, \%, c]$, $[\%, \langle acc, c \rangle, \%, \langle acc, c \rangle]$, where $c \in \{0, 1, B\}$.

(T4) For each instruction $\delta(p, a) = (q, b, R)$ of M', $[\%, b, 1p, \langle p, a \rangle]$, $[1p, \langle q, c \rangle, \%, c]$, where $c \in \{0, 1, B\}$.

(T5) For each instruction $\delta(p, a) = (q, b, L)$ of M', $[\%, \langle q, c \rangle, 0p, c]$, $[0p, b, \%, \langle p, a \rangle]$, where $c \in \{0, 1, B\}$.

Clearly the number of tiles in $T_{M'}$ does not depend on x.

Let us define the reduction f from BHP to BTP by $f(x) = \langle T_{M'}, 1^{p(|x|)}, 1^k, S_x \rangle$, where $\tilde{x}_L = u_1 u_2 \cdots u_k$, $u_i \in \{0, 1\}$, and

$$S_x = \{[\$, \langle q_0, u_1 \rangle, \#, \$], [\$, u_2, \#, \$], [\$, u_3, \#, \$] \ldots, [\#, u_k, \#, \$]\}.$$

The function f is one-one and p-invertible because so is h. We remark that, for each i with $2 \leq i \leq k$, the number of tiles t which matches $[\$, u_i, \#, \$]$ to the right (i.e., $left[t] = right[\$, u_i, \#, \$]$) is exactly 2, and consequently, the probability that the ith tile of S_x is chosen is $1/2$.

Claim 9 *The function f is a reduction from* BHP *to* BTP.

Proof of Claim. Assume that M' accepts a string $\tilde{x}_L$. By the choice of the initial row of length k, the sequence of the top of the tiles set in the bottom row of the square is exactly $\tilde{x}_L z$, where z satisfies $|\tilde{x}_L z| = p(|x|)$. This is an initial ID (i.e., instantaneous description) of the machine M' on input $\tilde{x}_L z$. The second row of tiles that match the bottom row represents the ID of M' on $\tilde{x}_L z$ obtained by a single application of the transition function δ (i.e., after one step) because a pair of tiles in (T4) (corresponding to $\delta(p_0, a) = (q, b, R)$) are the only tiles that are different from the symbols on the top of the bottom row. Recursively we can continue this argument, and then, when we rebuild the history of the ID's of M' on input $\tilde{x}_L z$, we see that the tiles fill the square. Recall that M' reaches the accepting state in less than $p(|x|)$ steps. Conversely, if the tiles fill the square, the sequence of the top of each row describes an ID of M' on input $\tilde{x}_L z$. This implies that M' accepts $\tilde{x}_L$. Hence, f becomes a reduction from BHP to BTP. ■

We should check the domination condition for f. Let $s(z)$ be a polynomial such that

$$s(z) \geq 1280 \cdot p(z)(p(z)+1)^2(q(z)+1)^2 \cdot \frac{\|T_{M'}\|}{\hat{\nu}(T_{M'})}$$

for all z. Then, we have:

$$\begin{aligned} s(|x|) \cdot \hat{\mu}_{\mathrm{BTP}}(f(x)) &= s(|x|) \cdot \hat{\mu}_{\mathrm{BTP}}(T_{M'}, 1^{p(|x|)}, S_x) \\ &= s(|x|) \cdot \hat{\nu}(T_{M'}) \cdot 2^{-2\mathrm{llog}(p(|x|))-1} \cdot \frac{1}{p(|x|)} \cdot \frac{1}{\|T_{M'}\|} \cdot \left(\frac{1}{2}\right)^{|\tilde{x}_L|-1} \\ &\geq s(|x|) \cdot \frac{\hat{\nu}(T_{M'})}{\|T_{M'}\|} \cdot \frac{1}{8p(|x|)(p(|x|)+1)^2} \cdot \frac{2}{2^{|\tilde{x}_L|}}. \end{aligned}$$

By the definition of s, the last term is bounded by:

$$\begin{aligned} 160 \cdot q(|x|+1)^2 \cdot \frac{2}{2^{2\mathrm{llog}(q(|x|))+|h(x)|+1}} &\geq 8 \cdot q(|x|+1)^2 \cdot \frac{20}{8(q(|x|)+1)^2} \cdot 2^{-|h(x)|} \\ &\geq 20 \cdot 2^{-|h(x)|} \\ &\geq \hat{\mu}_{\mathrm{BHP}}(x). \end{aligned}$$

Thus, $\hat{\mu}_{\mathrm{BTP}}(f(x)) \geq \frac{\hat{\mu}_{\mathrm{BHP}}(x)}{s(|x|)}$ for all x. It follows that $\mu_{\mathrm{BHP}} \preceq^{\mathrm{p}} \mu_{\mathrm{BTP}} \circ f$. The other direction $\mu_{\mathrm{BTP}} \preceq^{\mathrm{p}} \mu_{\mathrm{BHP}}$ is shown similarly using the inequality $\hat{\mu}_{\mathrm{BHP}}(x) \geq 4 \cdot 2^{-|h(x)|}$. Therefore, $\mu_{\mathrm{BHP}} \approx^{\mathrm{p}} \mu_{\mathrm{BTP}} \circ f$.

Next we define a reduction g from RBTP to RBHP. This part is simpler than the above argument. Again by the Distribution Controlling Lemma, there exists a total, one-one, p-invertible, **P**-computable function h' such that $4 \cdot 2^{-|h'(x)|} \leq \hat{\mu}_{\mathrm{BTP}}(x) \leq 20 \cdot 2^{-|h''(x)|}$ for all x. Take a nondeterministic Turing machine M which computes BTP in polynomial time. Let us define another Turing machine M' as follows:

begin algorithm M'
  input x
  compute w such that $x = h'(w)$
  if no such w exists **then** reject
  simulate M on input w
end.

Let p be a polynomial which is a time bound of M'. It is obvious that $x \in L(M)$ if and only if $h'(x) \in L(M')$ for all strings x. Let i be an index such that $L(M_i) = L(M')$. Let p be a polynomial such that $\mathrm{Time}_{M_i}(h(x)) \leq p(|x|)$ for all x. Define $g(x) = \langle s_i, h'(x), 1^{p(|x|)} \rangle$. Clearly g is one-one, length-increasing, **P**-computable, and p-invertible. Assume that $|h'(x)| \leq q(|x|)$ for some polynomial q. Take a polynomial s' so that

$$s'(z) \geq 1280 \cdot (p(z)+1)^2 \cdot (q(z)+1)^2 / \hat{\nu}_{\mathrm{stand}}(s_i).$$

The following calculation is similar to the one above.

$$\begin{aligned}
s'(|x|) \cdot \hat{\mu}_{\mathrm{BHP}}(g(x)) &= \hat{\mu}_{\mathrm{BHP}}(s_i, h'(x), 1^{p(|x|)}) \\
&= s'(|x|) \cdot \hat{\nu}_{\mathrm{stand}}(s_i) \cdot \hat{\nu}_{\mathrm{stand}}(h'(x)) \cdot \hat{\nu}_{\mathrm{tally}}(1^{p(|x|)}) \\
&\geq s'(|x|) \cdot \hat{\nu}_{\mathrm{stand}}(s_i) \cdot \frac{1}{8(q(|x|)+1)^2} \cdot \frac{1}{8(p(|x|)+1)^2} \cdot 2^{-|h'(x)|} \\
&\geq 20 \cdot 2^{-|h'(x)|} \\
&\geq \hat{\mu}_{\mathrm{BTP}}(x).
\end{aligned}$$

Hence, we have $\mu_{\mathrm{BTP}} \preceq^{\mathrm{p}} \mu_{\mathrm{BHP}} \circ g$. It is also easy to show that $\mu_{\mathrm{BHP}} \circ g \preceq^{\mathrm{p}} \mu_{\mathrm{BTP}}$. Therefore, $\mu_{\mathrm{BHP}} \circ g \approx^{\mathrm{p}} \mu_{\mathrm{BHP}}$. This completes the proof. □

Knijnenburg [51] considered complete problems for Dist(**PSPACE**, **P**-comp) and pointed out that an extension of randomized bounded tiling problem becomes $\leq^{\mathrm{p}}_m$-complete for Dist(**PSPACE**, **P**-comp).

5.3.3 Other Complete Problems

Gurevich [36] showed that the following randomized bounded Post correspondence problem is also $\leq^{\mathrm{p}}_m$-complete for Dist(**NP**, **P**-comp). The *randomized bounded Post correspondence problem* (RBPCP) is the distributional problem $(\mathrm{BPCP}, \mu_{\mathrm{BPCP}})$ defined as follows. Given a nonempty list $L = ((u_1, v_1), (u_2, v_2), \ldots, (u_m, v_m))$ of pairs of binary strings, the sequence $(i_1, i_2, \ldots, i_k)$, $1 \leq k \leq m$, is called a *solution* of length k if $u_{i_1} u_{i_2} \cdots u_{i_k} = v_{i_1} v_{i_2} \cdots v_{i_k}$. The set BPCP is defined by

$$\mathrm{BPCP} = \{\langle \langle L \rangle, 1^n \rangle \mid L = ((u_1, v_1), \ldots, (u_m, v_m)) \wedge \exists k \leq n [\text{ there exists a solution of length } k \text{ for } L\ ]\},$$

where $\langle L \rangle$ is the encoding $\langle \langle u_1, v_1 \rangle, \ldots, \langle u_m, v_m \rangle \rangle$, and the distribution μ_{BPCP} is defined by

$$\hat{\mu}_{\mathrm{BPCP}}(\langle \langle u_1, v_1 \rangle, \ldots, \langle u_m, v_m \rangle, 1^n \rangle) = \hat{\nu}_{\mathrm{tally}}(1^n) \cdot \hat{\nu}_{\mathrm{tally}}(1^m) \cdot \prod_{i=1}^{s} (\hat{\nu}_{\mathrm{stand}}(u_i) \cdot \hat{\nu}_{\mathrm{stand}}(v_i)).$$

This default probability is experimentally given by picking up independently and randomly two natural numbers n and m, and $2m$ strings $u_1, \ldots, u_m, v_1, \ldots, v_m$.

This problem RBHP is p-m-reducible to the randomized bounded Post correspondence problem RBPCP. We omit the proof; the interested reader may refer to [36].

By a slight modification of RBPCP, Gurevich [35] introduced the *Randomized Bounded String Correspondence Problem* and showed that this problem is also p-m-complete for Dist(**NP**, **P**-comp).

Another problem known to be complete is the *Randomized Palindrome Problem*, which was defined by Gurevich [36]. Wang and Belanger [112] introduced the *Randomized Word Problem for Thue System* and showed that this problem is p-m-complete for Dist($\mathbf{NP}, \mathbf{P}$-comp).

5.3.4 Hard Problems under Samplable Distributions

We have seen several intriguing complete problems for Dist($\mathbf{NP}, \mathbf{P}$-comp). A natural question is: does the class Dist($\mathbf{NP}, \mathbf{P}$-samp) have any p-m-complete problems ? Ben-David, Chor, Goldreich, and Luby [9] constructed such a distributional problem.

First of all, we shall define a "universal" $\mathbf{P}$-samplable distribution μ_U. Take an effective enumeration $\{\eta_i\}_{i\in\mathbb{N}}$ of all $O(n)$-time samplable distributions. Let us define μ_U as follows:

$$\hat{\mu}_U(x) = \sum_{i=0}^{\infty} 2^{-2\mathrm{llog}(i)-1} \cdot \hat{\eta}_i(x).$$

Experimentally, we choose a number i at random, then sample a string under the distribution η_i. Clearly μ_U is a $\mathbf{P}$-samplable distribution.

Theorem 5.3.4 **[9]** *The distributional decision problem* (BHP, μ_U) *is p-m-complete for* Dist($\mathbf{NP}, \mathbf{P}$-samp).

Proof. WE must reduce an arbitrary distributional problem (D, μ) in Aver($\mathbf{NP}, \mathbf{P}$-samp) to (BHP, μ_U).

By Lemma 4.4.3, we can take a strictly $\mathbf{P}$-samplable distribution ν such that $\mu \preceq^{\mathrm{p}} \nu$. By its samplability, there exists a randomized Turing machine M sampling ν in time q in the length of its output, where q is an appropriate polynomial. For each x, let $g(x)$ be $x0^{q(|x|)-|x|}$. Let us define ν' by $\nu_{g^{-1}}$. We show that this distribution ν' becomes $O(n)$-samplable. To sample ν', we consider the following procedure M': on input λ, simulate M, and if x is an output of M, then output $x0^{q(|x|)-|x|}$. We fix any random seed s which leads to the output x. The running time $\mathrm{Time}_{M'}(\lambda; s)$ is $O(\mathrm{Time}_M(\lambda; s) + q(|x|)) \subseteq O(q(|x|))$ because $\mathrm{Time}_M(\lambda; s) \leq q(|x|)$. As a result, it is bounded above by $O(n)$ in the length n of output.

We then define the set D' by

$$D' = \{y \mid \exists x, w[y = xw \wedge x \in D \wedge q(|x|-1) < |y| \leq q(|x|)]\}.$$

Obviously, D' is in $\mathbf{NP}$ since D is so. By the definition of g, it follows that $x \in D$ if and only if $g(x) \in D'$, and thus $(D, \mu) \leq_m^{\mathrm{p}} (D', \nu')$ via g.

Now we must show that (D', ν') is p-m-reducible to (BHP, μ_U). Because D is an $\mathbf{NP}$ set, it is recognizable by some nondeterministic Turing machine, say M, in polynomial time. Let p be a strictly increasing polynomial which bounds the running time of M. Let i be an index so that $L(M_i) = L(M)$ in the list $\{M_i\}_{i\in\mathbb{N}}$. We take the standard reduction function f discussed before: $f(x) = \langle s_i, x, 1^{p(|x|)}\rangle$ for all x.

Let $\mathrm{Time}_f(x)$ be the computation time needed to compute $f(x)$ deterministically. Since f is $\mathbf{P}$-computable, for some constants $c, c' > 0$, $\mathrm{Time}_f(x)$ is bounded by

$$\mathrm{Time}_f(x) \leq c \cdot (|x| + p(|x|) + 1) \leq c' \cdot (\langle s_i, x, 1^{p(|x|)}\rangle + 1) = c' \cdot |f(x)| + c'.$$

Thus, f is computable in $O(n)$ steps in the length n of its output.

Now define $\hat{\eta}(x) = \hat{\nu}'(f^{-1}(x))$ for all x. This η is $O(n)$-time samplable, and thus there exists an index i such that $\eta = \eta_i$. By the definition of μ_U, $\hat{\mu}_U(x) \geq 2^{-2\mathrm{llog}(x)-1} \cdot \hat{\eta}_i(x)$. Then, we have:

$$\hat{\mu}_U(x) \geq \frac{1}{8(i+1)^2} \cdot \hat{\nu}'(x).$$

Therefore, (A, μ) is p-m-reducible to (BHP, μ_U). □

5.3.5 Discussion of Complete Problems for Aver($\mathbf{NP}, \mathbf{P}$-comp)

We have seen several complete problems for Dist($\mathbf{NP}, \mathbf{P}$-comp) and Dist($\mathbf{NP}, \mathbf{P}$-samp). In this subsection, we shall discuss complete problems for the average-case complexity class Aver($\mathbf{NP}, \mathbf{P}$-comp).

Unfortunately, it is not known whether Aver($\mathbf{NP}, \mathbf{P}$-comp) has any $\leq_m^{\mathrm{p}}$-complete problems. This is because of our definition of nondeterministic Turing machines and their accepting criteria. In this section, we shall see what happens when we take other models of nondeterministic Turing machines and accepting criteria.

However, by Wang and Belanger [111], if we use a "clocked" model of nondeterministic Turing machines (thus, all computation paths are assumed to be of the same length), then any p-m-complete problem for Dist($\mathbf{NP}, \mathbf{P}$-comp) is $\leq_m^{\mathrm{avp}}$-complete for Aver($\mathbf{NP}^*, \mathbf{P}$-comp). Recall the notation Aver($\mathbf{NP}^*, \mathcal{F}$) used in Section 3.6. We shall see a complete problem for Aver($\mathbf{NP}^*, \mathbf{P}$-comp).

We begin with a general result. Let us expand a notion of "weak $\mathcal{C}$-descriptive" to "$\leq_\alpha$-descriptive" using any $\leq_\alpha$-reducibility.

Definition 5.3.5 ($\leq_\alpha$-Descriptive) Let $\leq_\alpha$ be any reducibility. An average complexity class Aver($\mathcal{D}, \mathcal{F}$) is *$\leq_\alpha$-descriptive* if, for every problem (D, μ) in Aver($\mathcal{D}, \mathcal{F}$), there exists a problem (E, ν) in Dist($\mathcal{D}, \mathcal{F}$) such that $(D, \mu) \leq_\alpha (E, \nu)$.

Proposition 5.3.6 *Let $\mathcal{C}$ be a complexity class and let $\mathcal{F}$ be a set of distributions. Assume that* Aver($\mathcal{C}, \mathcal{F}$) *is defined and $\leq_\alpha$-descriptive for reducibility $\leq_\alpha$. Moreover, assume that $\leq_m^{\mathrm{p}}$ implies $\leq_\alpha$, and $\leq_\alpha$ is transitive. If a distributional problem (A, μ) is $\leq_m^{\mathrm{p}}$-complete for* Dist($\mathcal{C}, \mathcal{F}$), *then (A, μ) is also $\leq_\alpha$-complete for* Aver($\mathcal{C}, \mathcal{F}$).

Proof. Assume that (A, μ) is $\leq_m^{\mathrm{p}}$-complete for Dist($\mathcal{C}, \mathcal{F}$). Let (B, ν) be an arbitrary distributional problem in Aver($\mathcal{C}, \mathcal{F}$). We shall show that (B, ν) is $\leq_\alpha$-reducible to (A, ν). Since Aver($\mathcal{C}, \mathcal{F}$) is $\leq_\alpha$-descriptive, there is a problem $(B', \nu') \in$ Dist($\mathcal{C}, \mathcal{F}$) such that (B, ν) is $\leq_\alpha$-reducible to (B', ν'). From the assumption that (A, μ) is $\leq_m^{\mathrm{p}}$-complete for Dist($\mathcal{C}, \mathcal{F}$), it follows that $(B', \nu') \leq_\alpha (A, \mu)$. The transitivity of $\leq_\alpha$ implies that $(B, \nu) \leq_\alpha (A, \mu)$. □

Claim 10 RBHP *and* RBTP *are avp-m-complete for* Aver($\mathbf{NP}^*, \mathbf{P}$-comp).

Proof of Claim. It suffices to show that Aver($\mathbf{NP}^*, \mathbf{P}$-comp) is $\leq_m^{\mathrm{avp}}$-descriptive. Assume that $(A, \mu) \in$ Aver($\mathbf{NP}^*, \mathbf{P}$-comp). Let M be a nondeterministic Turing machine which computes A in time polynomial on μ-average. Let us take g as in Lemma 4.2.7(2). Note that g is $\mathbf{P}$-computable and p-invertible. Then, we have $\hat{\mu}(x) < 2^{-|g(x)|+2}$ for all strings x. We define ν as

$$\hat{\nu}(z) = \begin{cases} \hat{\nu}_{\mathrm{stand}}(g(x)) \cdot \hat{\nu}_{\mathrm{tally}}(1^n) & \text{if } z = \langle x, 1^n \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

It is not difficult to see that ν is a $\mathbf{P}$-computable distribution and does not depend on μ.

To get the desired result, we would define a many-one reduction f and a set B in $\mathbf{NP}$. Let $f(x) = \langle g(x), 1^{\mathrm{Time}_M(x)} \rangle$ for every x. We claim that $(f, \mu) \in$ Aver($\mathbf{FP}, \mathbf{P}$-comp). Note that the function $\lambda x.\mathrm{Time}_M(x)$ is computable in time polynomial on μ-average. Since $g \in \mathbf{FP}$, $f(x)$ is computable in time polynomial on μ-average. Note that f is one-one. Let $B = f(A)$. It follows that $B \in \mathbf{NP}$ since B is computed by the following Turing machine M':

begin nondeterministic algorithm for M'
  input $\langle x, 1^n \rangle$
  compute u for which $f(u) = x$
  simulate M on input u for time n
  if M fails to halt **then** reject
  output $M(x)$ and **halt**
end.

This implies that $(B, \nu) \in \mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. Now we set $q(x) = 256(|g(x)|+1)^2(\mathrm{Time}_M(x)+1)^2$. Then, by Lemma 3.3.14, q turns out to be polynomial on μ-average.

We next show that $\hat{\nu}(f(x)) \cdot q(x) \geq \hat{\mu}(x)$. For all x, we have

$$\begin{aligned} q(x) \cdot \hat{\nu}(f(x)) &= q(x) \cdot \hat{\nu}_{\mathrm{stand}}(g(x)) \cdot 2^{-2\mathrm{llog}(\mathrm{Time}_M(x))-1} \\ &\geq \frac{q(x)}{64(|g(x)|+1)^2(\mathrm{Time}_M(x)+1)^2} \cdot 2^{-|g(x)|} \\ &= \frac{q(x)}{256(|g(x)|+1)^2(\mathrm{Time}_M(x)+1)^2} \cdot 2^{-|g(x)|+2} \\ &= 2^{-|g(x)|+2} \quad > \quad \hat{\mu}(x). \end{aligned}$$

Therefore, $\mu_{f^{-1}} \preceq^{\mathrm{p}} \nu$. ∎

If we use the length of the longest computation path whenever the machine rejects an input as the running time, then any p-m-complete problem for Dist($\mathbf{NP}, \mathbf{P}$-comp) is avp-T-complete for Aver($\mathbf{NP}^*, \mathbf{P}$-comp).

Claim 11 RBHP *and* RBTP *are avp-T-complete for* Aver($\mathbf{NP}^*, \mathbf{P}$-comp)*, where* Aver($\mathbf{NP}^*, \mathbf{P}$-comp) *is defined based on the model of nondeterministic Turing machines with running time measured by the longest rejecting path whenever it rejects an input.*

Proof of Claim. For this claim, it suffices to show that Aver($\mathbf{NP}^*, \mathbf{P}$-comp) is $\leq_T^{\mathrm{avp}}$-descriptive. Assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}^*, \mathbf{P}\text{-comp})$. Let M be a nondeterministic Turing machine which computes A in time polynomial on μ-average. Assume that g is defined as in Lemma 4.2.7(2). Note that g is $\mathbf{P}$-computable and p-invertible. Then, we have $\hat{\mu}(x) < 2^{-|g(x)|+2}$ for all strings x.

Recall the proof of Claim 3 in Section 3.6, and consider the same set C and the deterministic oracle Turing machine N that computes D with oracle C in polynomial time on μ-average. Remember that, for each query string $\langle b, x, 1^n \rangle$, x is the only input string that N on input x can query $\langle b, x, 1^n \rangle$ to oracle C. Let us define ν as

$$\hat{\nu}(z) = \begin{cases} \frac{1}{2} \cdot \hat{\nu}_{\mathrm{stand}}(x) \cdot \hat{\nu}_{\mathrm{tally}}(1^n) & \text{if } z = \langle b, x, 1^n \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

It is not difficult to see that ν is a $\mathbf{P}$-computable distribution and does not depend on μ or C.

We modify the Turing machine N into N' in the following fashion:

begin algorithm N'
  input x
  compute u for which $g(u) = x$
  simulate N with oracle C on input u
  output $N'(x)$ and **halt**
end.

This implies that $(B, \nu) \in \mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. Now we set $q(x) = 512(|g(x)|+1)^2(\mathrm{Time}_{N'}^C(x)+1)^2$. Then by Lemma 3.3.14, q turns out to be polynomial on μ-average.

For all x, we have the simple estimation, where $n = \mathrm{Time}_{N'}^C(x)$,

$$\begin{aligned} q(x) \cdot \hat{\nu}(b, g(x), 1^n) &= q(x) \cdot \frac{1}{2} \cdot \hat{\nu}_{\mathrm{stand}}(g(x)) \cdot \hat{\nu}_{\mathrm{tally}}(1^n) \\ &\geq \frac{q(x)}{512(|g(x)|+1)^2(\mathrm{Time}_{N'}^C(x)+1)^2} \cdot 2^{-|g(x)|+2} \\ &= 2^{-|g(x)|+2} > \hat{\mu}(x). \end{aligned}$$

Therefore, $(D, \mu) \leq_T^{\mathrm{avp}} (C, \nu)$ via N'. ■

5.4 Incompleteness Results

We shall discuss two important distributions: flat distributions and sparse distributions, both of which possibly make associated distributional problems incomplete for Dist($\mathbf{NP}, \mathbf{P}$-comp).

5.4.1 Flat Distributions

Let us recall the definition of flat distributions. A distribution μ is called *flat* if there exists a real number $\epsilon > 0$ such that $\hat{\mu}(x) \leq 2^{-|x|^\epsilon}$ for almost all x.

We shall observe the distributions of some p-m-complete problems discussed in the previous section. Neither μ_{BHP}, μ_{BTP}, nor μ_{BPCP}, for example, is a flat distribution. Here a distribution μ is flat if there exists a real number $\epsilon > 0$ such that $\hat{\mu}(x) \leq 2^{-|x|^\epsilon}$ for all x. This is seen as follows. For example, assuming that μ_{BHP} is flat (i.e., $\hat{\mu}_{\mathrm{BHP}}(x) \leq 2^{-|x|^\epsilon}$ for some $\epsilon > 0$), let us consider sufficiently large n, i, and x satisfying $n \geq (\mathrm{llog}(i) + |x|)^{4/\epsilon}$. Then we have

$$\begin{aligned}
\hat{\mu}_{\mathrm{BHP}}(s_i, x, 1^n) &= 2^{-2\mathrm{llog}(|s_i|)-|s_i|-1} \cdot 2^{-2\mathrm{llog}(|x|)-|x|-1} \cdot 2^{-2\mathrm{llog}(n)-1} \\
&\geq 2^{-\mathrm{llog}(i)-|x|} \cdot \frac{1}{8(\mathrm{llog}(i)+1)^2} \cdot \frac{1}{8(|x|+1)^2} \cdot \frac{1}{8(n+1)^2} \\
&\geq \frac{2^{-\mathrm{llog}(i)-|x|}}{512} \cdot \frac{1}{4\mathrm{llog}(i)^2 \cdot 4|x|^2 \cdot 4n^2} \\
&> \frac{2^{-\mathrm{llog}(i)-|x|}}{n^3 \cdot \mathrm{llog}(i)^2 \cdot |x|^2}.
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
|\langle s_i, x, 1^n \rangle| &\geq 2(\mathrm{llog}(i) + |x|) + n \\
&\geq n^{1/2} \cdot (\mathrm{llog}(i) + |x|)^{2/\epsilon} \left(\frac{2(\mathrm{llog}(i)+|x|)}{(\mathrm{llog}(i)+|x|)^{2/\epsilon} \cdot n^{1/2}} + \frac{n}{n^{1/2} \cdot (\mathrm{llog}(i)+|x|)^{2/\epsilon}} \right) \\
&\geq n^{1/2} \cdot (\mathrm{llog}(i) + |x|)^{2/\epsilon} \cdot \frac{2(\mathrm{llog}(i)+|x|)+n}{n^{1/2}n^{1/2}} \\
&> n^{1/2} \cdot (\mathrm{llog}(i) + |x|)^{2/\epsilon}.
\end{aligned}$$

Hence, using the assumption that μ_{BHP} is flat,

$$\hat{\mu}_{\mathrm{BHP}}(s_i, x, 1^n) \leq 2^{-|\langle s_i, x, 1^n \rangle|^\epsilon} < 2^{-n^{\epsilon/2}(\mathrm{llog}(i)+|x|)^2} < 2^{-n^{\epsilon/2}-2\mathrm{llog}(i)-2|x|}.$$

Now we have $\frac{2^{-\mathrm{llog}(i)-|x|}}{n^3 \mathrm{llog}(i)^2 |x|^2} < 2^{-n^{\epsilon/2}-2\mathrm{llog}(i)-2|x|}$, and thus $2^{\mathrm{llog}(i)+|x|+n^{\epsilon/2}} < n^3 \cdot \mathrm{llog}(i)^2 \cdot |x|^2$. This is clearly a contradiction. Hence, μ_{BHP} is not flat.

One might suspect that there is no $\leq^{\mathrm{p}}_m$-complete problem with a flat distribution. Gurevich [36], and Wang and Belanger [112] indeed showed that distributional problems with flat distributions are not complete for Dist($\mathbf{NP}, \mathbf{P}$-comp).

Using the same argument as above, we can show the following result. Under p-honest, $\mathbf{P}$-computable, one-one reductions, there are no complete problems with flat distributions in Dist($\mathbf{NP}, \mathbf{P}$-comp).

Proposition 5.4.1 **[112]** *For any set $D \in \mathbf{NP}$ and any flat distribution μ, the distributional problem (D, μ) cannot be $\leq^{\mathrm{p}}_1$-complete for* Dist($\mathbf{NP}, \mathbf{P}$-comp) *under one-one, p-honest reductions.*

Proof. We shall prove the contrapositive. Assume that (A, μ) is $\leq^{\mathrm{p}}_1$-complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) under one-one, p-honest reductions. As RBHP is in Dist($\mathbf{NP}, \mathbf{P}$-comp), there is a p-honest, $\mathbf{P}$-computable, one-one reduction f that reduces RBHP to (A, μ). The domination condition for f, by Lemma 3.4.12(1), implies that $\mu_{\mathrm{BHP}} \preceq^{\mathrm{p}} \mu \circ f$ since f is one-one. Since μ_{BHP} is not a flat distribution, then $\mu \circ f$ is also non-flat.

As f is p-honest, μ cannot be flat. To see this, assume that $\hat{\mu}(x) \leq 2^{-|x|^\epsilon}$ and $|f(x)|^k \leq |x|$ for $\epsilon, k > 0$. Thus, $\widehat{\mu \circ f}(x) \leq 2^{-|f(x)|^\epsilon} \leq 2^{-|x|^{\epsilon/k}}$. □

Proposition 5.4.2 **[36]** *Assume that* **EXP** $\neq$ **NEXP**. *For any flat distribution μ and any* **EXP** *set D, the distributional problem (D, μ) is not avp-m-hard for* Dist(**NP**, **P**-comp).

Proof. Assume that (D, μ) is $\leq_m^{\mathrm{avp}}$-hard for Dist(**NP**, **P**-comp) and show that any set in **NEXP** is deterministically solvable in exponential time.

Let A be an arbitrary set in **NEXP**. There is a nondeterministic Turing machine M which computes A in time $2^{p(|x|)}$, where p is a fixed polynomial. We may assume that $p(n) > n + 2\mathrm{llog}(n) + 1$ for all $n \in \mathbb{N}$. For each x, set $x' = x01^{2^{p(|x|)}-|x|-1}$. We note that if $|x| = n$, then $|x'| = 2^{p(n)}$. We set $A' = \{x' \mid x \in A\}$. This A' belongs to **NP** and consequently it is in **EXP**. Let $M_{A'}$ be a deterministic Turing machine which computes A' in exponential time.

We define ν' as follows:

$$\hat{\nu}'(z) = \begin{cases} \hat{\nu}_{\mathrm{stand}}(x) & \text{if } z = x' \text{ for some } x, \\ 0 & \text{otherwise.} \end{cases}$$

To see that ν' is **P**-computable, notice that $\nu'(z) = \nu_{\mathrm{stand}}(\max\{u \mid u01^{2^{p(|u|)}-|u|-1} \leq z\})$, and thus $\lim_{z\to\infty} \nu(z) = \lim_{u\to\infty} \nu_{\mathrm{stand}}(u) = 1$.

Since $(A', \nu') \in \mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$, our assumption implies that (A', ν') is avp-m-reducible to (D, μ). Take a reduction f witnessing $(A', \nu') \leq_m^{\mathrm{avp}} (D, \mu)$. Note that f is computable in time polynomial on ν'-average, and $\nu' \preceq^{\mathrm{avp}} \eta$ and $\hat{\mu} \geq \hat{\eta}_{f^{-1}}$ for some semi-distribution η. We note that, by the definition of ν', f is actually computable in exponential time.

First we show that $|f(x')|$ is bounded by a polynomial in $|x|$. The domination condition yields the existence of a function s being polynomial on ν'-average such that $\hat{\mu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\nu}'(x)}{s(x)}$ for all y. In particular, we have $\hat{\mu}(f(x')) \geq \frac{\hat{\nu}'(x')}{s(x')}$. Assume that s is q on ν'-average for some polynomial q. Then, for almost all x,

$$\begin{aligned} s(x') &\leq q(|x'|/\hat{\nu}'(x')) = q(2^{p(|x|)} \cdot 2^{2\mathrm{llog}(|x|)+|x|+1}) \\ &\leq q(2^{2p(|x|)}) \leq 2^{k \cdot p(|x|)} \end{aligned}$$

for some fixed constant $k \geq 2$. Thus, $s(x') \leq 2^{k \cdot p(|x|)}$. Using this inequality, we have

$$\hat{\mu}(f(x')) \geq \frac{\hat{\nu}'(x')}{s(x')} \geq 2^{-k \cdot p(|x|)} \cdot 2^{-2\mathrm{llog}(|x|)-|x|-1} \geq 2^{-(k+1)p(|x|)}$$

for almost all x. Since μ is flat, for some constant $m > 0$, we have

$$2^{-|f(x')|^{1/m}} \geq \hat{\mu}(f(x')) \geq 2^{-(k+1)p(|x|)}.$$

This yields the desired consequence that $|f(x')| \leq (k+1)^m \cdot p(|x|)^m$.

Let us consider the following deterministic algorithm M which computes A.

begin deterministic algorithm M
  input x (say, $n = |x|$)
  compute x' $(= x01^{2^{p(n)}-n-1})$
  compute $f(x')$
$(*)$ simulate $M_{A'}$ on input $f(x')$ and halt
end.

The running time of line $(*)$ is at most exponential in $|x|$ since $|f(x')|$ is bounded by a polynomial in $|x|$. Hence, the total running time of algorithm M is exponential in the length of input. This implies that A is in **EXP**. □

5.4.2 Sparse Distributions

Gurevich [36] called a distributional problem (D, μ) "sparse" if the set $\{x \mid \hat{\mu}(x) > 0\}$ is sparse, where a set S is *(polynomially) sparse* if there is a polynomial p such that $\|S \cap \Sigma^n\| \leq p(n)$ for all $n \in \mathbb{N}$. In this section, we instead call the distribution μ *(polynomially) sparse* if the set $\{x \mid \hat{\mu}(x) > 0\}$ is sparse. In other words, μ is positive only on a sparse set. For instance, the standard distribution on $\{0\}^*$ is sparse.

Definition 5.4.3 (Sparse Distributions) A distribution μ is called *(polynomially) sparse* if the set $\{x \mid \hat{\mu}(x) > 0\}$ is sparse.

We shall see that any distributional problems having a flat distribution cannot be p-isomorphic to the standard complete problem RBHP.

Lemma 5.4.4 *For any set D in* **NP** *and any* **P***-computable distribution μ, if μ is sparse, then (D, μ) is not p-isomorphic to* RBHP.

Proof. Assume that (D, μ) is p-isomorphic to RBHP via a bijection f. The domination condition by Lemma 3.4.12(1) implies that there exists a p-bounded function p such that $p(x) \cdot \hat{\mu}(f(x)) \geq \hat{\mu}_{\mathrm{BHP}}(x)$. For the sake of convenience, we set $S = \{\langle s_i, x, 1^n \rangle \mid i, n \in \mathbb{N}, x \in \Sigma^*\}$.

Since μ_{BHP} is positive on S, the distribution $\mu \circ f$ is also positive on S; that is,

$$f(S) = \{f(x) \mid x \in S\} \subseteq \{z \mid \hat{\mu}(z) > 0\}.$$

By the sparseness of the set $\{z \mid \hat{\mu}(z) > 0\}$, the set $f(S)$ is also sparse. In particular, $f(\mathrm{BHP})$ is sparse. Therefore, since f is one-one, BHP should be sparse. This contradicts the fact that BHP is not sparse. □

It is known that, under the isomorphism conjecture, no sparse sets can be **NP**-complete. In 1988, Mahaney [66] proved that, without assuming the isomorphism conjecture, there is no sparse **NP**-complete set unless $\mathbf{P} = \mathbf{NP}$. Notice that the isomorphism conjecture conflicts with the assumption $\mathbf{P} \neq \mathbf{NP}$. A

similar result also holds in our average-case environment.

Theorem 5.4.5 *Assume that* $\mathbf{P} \neq \mathbf{NP}$. *For any set* D *and any sparse distribution* μ, *the distributional problem* (D, μ) *is not p-m-hard for* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$.

Proof. Assuming that (D, μ) is p-m-hard for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ and μ is sparse, we shall show that **NP** collapses to **P**. In particular, $(\mathrm{BHP}, \mu_{\mathrm{BHP}}) \leq_m^{\mathrm{p}} (D, \mu)$ via some reduction f. By an argument similar to that in the proof of Lemma 5.4.4, the domination condition by Lemma 3.4.12(1) implies that there exists a p-bounded function p such that $\hat{\mu}(z) \geq \sum_{x \in f^{-1}(z)} \frac{\hat{\mu}_{\mathrm{BHP}}(x)}{p(x)}$ for all z. Thus, in particular, it holds that $p(x) \cdot \hat{\mu}(f(x)) \geq \hat{\mu}_{\mathrm{BHP}}(x)$. As in Lemma 5.4.4, the set $f(S)$ is sparse, where $S = \{\langle s_i, x, 1^n \rangle \mid i, n \in \mathbb{N}, x \in \Sigma^*\}$.

We shall use the fact that BHP is "self-reducible," that is, to determine whether $\langle s_i, x, 1^n \rangle \in \mathrm{BHP}$, we can check whether $\langle s_{j_0}, x, 1^{n-1} \rangle \in \mathrm{BHP}$ or $\langle s_{j_1}, x, 1^{n-1} \rangle \in \mathrm{BHP}$, where j_0 is an index of the machine that deterministically chooses the first nondeterministic branch and then simulates M_i, and j_1 is similarly defined by choosing 1 instead of 0. We view this process as a tree. Since we can determine whether $\langle s_i, x, 1^n \rangle \in \mathrm{BHP}$ within n steps, the height of the tree is n. Now let us consider such a self-reduction tree. This tree may contain exponentially-many nodes, but when they are mapped by f, there are at most polynomially-many distinct values taken by f. Hence, many nodes of the tree merge. This motivates us to construct a polynomial-time algorithm which computes BHP in the following fashion.

Let us first describe the main body of the algorithm M below:

begin deterministic algorithm M for BHP
  input $\langle s_i, x, 1^n \rangle$
  set $Visit := \emptyset$ and set $Dead := \emptyset$
  call $Marking(\langle s_i, x, 1^n \rangle, Visit, Dead)$
  reject and **halt**
end.

$Marking(w, Visit, Dead)$ is the following subroutine that recursively kills nodes which do not lead to an accepting configuration in a depth-first search:

subroutine $Marking(w, Visit, Dead)$
  if w is a leaf and true **then** accept and **halt**
  set $Visit := Visit \cup \{w\}$
  compute $f(w)$
  for all nodes $u \in Vist$ **until** neither $Visit$ nor $Dead$ changes
    if $f(w) = f(u)$ and $u \in Dead$ **then** set $Dead := Dead \cup \{w\}$
    if both children u_0, u_1 of u are in $Dead$
      then $Dead := Dead \cup \{u\}$ and $Visit := Visit - \{u_0, u_1\}$
    if u is a leaf and false **then** $Dead := Dead \cup \{u\}$
  end-for

construct a left child of w, say w_0
call $Marking(w_0, Visit, Dead)$
construct a right child of w, say w_1
call $Marking(w_1, Visit, Dead)$
return.

The algorithm M requires at most polynomially-bounded running time and also computes BHP. Therefore, BHP belongs to **P**. The conclusion **P** = **NP** follows immediately from the fact that BHP is p-m-complete for **NP**. □

5.4.3 Unreasonable Distributions

In 1996, Paven and Selman [83] presented another type of incompleteness result. They called a distribution μ *reasonable* if $\lambda n.\hat{\mu}(\Sigma^{\geq n}) \in \Omega(n^{-k})$ for some number $k > 0$. Any distributional decision problem with an *un*reasonable distribution fails to be hard for Dist(**NP**, **P**-comp) unless **NP** is *small* (i.e., **NP** has p-measure 0). Since many researchers believe that **NP** is not small, this result shows another limitation of distributions in distributional complete problems.

Theorem 5.4.6 **[83]** *Assume that* **NP** *has p-measure 1. Let D be a set and let μ be such that $\lambda n.\hat{\mu}(\Sigma^{\geq n}) \in \Omega(n^{-k})$ for any positive integer k. Then, the distributional problem (D, μ) is not p-m-hard for* Dist(**NP**, **P**-comp).

Proof. Assume that $\lambda n.\hat{\mu}(\Sigma^{\geq n}) \in \Omega(n^{-k})$ for any positive integer k. We also assume that (D, μ) is p-m-hard for Dist(**NP**, **P**-comp). In particular, for any **NP** set A, the distributional problem $(A, \nu_{\mathrm{stand}})$ is p-m-reducible to (D, μ). Since $D \in$ **EXP**, there exist two positive integers k' and c such that D belongs to DTIME$(c \cdot 2^{n^{k'}} + c)$. For brevity, we can assume $k' = k$. Note that this assumption does not essentially affect the following argument.

Let $\{f_i\}_{i \in \mathbb{N}}$ be an effective enumeration of all polynomial-time computable functions. Take any set A in **NP** and assume that $(A, \nu_{\mathrm{stand}}) \leq_m^{\mathrm{p}} (D, \mu)$ via f_m. We claim the following.

Claim 12 *There exist infinitely-many strings x such that $|f_m(x)|^k \leq |x|$.*

Proof of Claim. Assume to the contrary that there exist positive integers k and n_0 such that $|f_m(x)|^k > |x|$ for all strings x of length $\geq n_0$. We shall show that μ satisfies $\lambda n.\hat{\mu}(\Sigma^{\geq n}) \in \Omega(n^{-d})$ for some $d > 0$. This clearly leads to a contradiction.

By the domination condition for the reduction function f_m, there exists a semi-distribution η such that $\nu_{\mathrm{stand}} \preceq^{\mathrm{p}} \eta$ and $\hat{\mu} \geq \hat{\eta}_{f^{-1}}$. Take a positive polynomial q such that $q(|x|) \cdot \hat{\eta}(x) \geq \hat{\nu}_{\mathrm{stand}}(x)$ for all x. This

implies that $\hat{\eta}(\Sigma^n) \geq \hat{\nu}_{\mathrm{stand}}(\Sigma^n)/q(n)$ for any integer n.

$$\hat{\eta}(\Sigma^{\geq n}) \geq \sum_{i \geq n} \frac{\hat{\nu}_{\mathrm{stand}}(\Sigma^i)}{q(i)} \geq \sum_{i \geq n} \frac{1}{8q(i)(i+1)^2} \geq \frac{1}{8q(n) \cdot 4n^2} = \frac{1}{n^3 q(n)} \geq \frac{1}{n^{d+3}}$$

for any number $n \geq 32$.

Fix n arbitrarily but greater than $n_0^{1/k}$. Then,

$$\begin{aligned} \hat{\mu}(\Sigma^{\geq n}) &\geq \hat{\mu}\{z \in f_m(\Sigma^*) \mid |z| \geq n\}) \geq \hat{\eta}_{f^{-1}}(\{z \in f_m(\Sigma^*) \mid |z| \geq n\}) \\ &= \hat{\eta}(\{w \mid \exists z \in \mathrm{ran}(f_m) \cap \Sigma^{\geq n}[f_m(w) = z]\}) \\ &\geq \hat{\eta}(\Sigma^{\geq n^{1/k}}) \geq \frac{1}{n^{(d+3)/k}}. \end{aligned}$$

Now let $s = (d+3)/k + 1$. The above inequality implies that $\lambda n.\hat{\mu}(\Sigma^{\geq n}) \in \Omega(n^{-s})$. ■

Recall that $D \in \mathrm{DTIME}(c \cdot 2^{n^k} + c)$. We define the 1-MS d as follows:

$$d(m,z) = \begin{cases} 1 & \text{if } z = \lambda, \\ d(m,w) & \text{if } |f_m(s_{|z|})|^k > |s_{|z|}|, \\ 2d(m,w) & \text{if } |f_m(s_{|z|})|^k \leq |s_{|z|}| \text{ and } b = [f_m(s_{|z|}) \in D], \\ 0 & \text{if } |f_m(s_{|z|})|^k \leq |s_{|z|}| \text{ and } b \neq [f_m(s_{|z|}) \in D], \end{cases}$$

where $b \in \{0,1\}$ and $z = wb$. It is easy to check that d is **P**-computable. Let us check if d succeeds on D. By the definition of d, $d_m(D[0..n]) = 2d_m(D[0..n-1])$, where $d_m(w) = d(m,w)$. Thus, we obtain $\limsup_{n\to\infty} d_m(D[0..n]) = \infty$, and consequently d succeeds on D.

For each $i \in \mathbb{N}$, we define the set A_i to be the set on which d_i succeeds. Notice that the collection $\{A_i\}_{i\in\mathbb{N}}$ is a p-union of all **NP** sets. We thus conclude by Lemma2.7.9 that **NP** has p-measure 0. This contradicts our assumption. □

5.5 Bounded-Error Probabilistic Reducibility

As seen in Section 5.4, deterministic many-one reductions are so restricted that, under the common belief that **EXP** $\neq$ **NEXP**, no distributional problems with flat distributions are complete for Dist(**NP**, **P**-comp). Many "natural" distributions for graph-related problems are actually flat. Is there any hope that we can prove such problems to be "hard" to compute ? Venkatesan and Levin [106] presented a solution by introducing another reducibility, called "random many-one reducibility," to measure the complexity of distributional problems. Their notion of "random many-one reducibility" was further studied by Impagliazzo and Levin [44] and extended by Blass and Gurevich [12] (see Section 5.7).

From a different point of view, Ben-David, Chor, Goldreich, and Luby [9] introduced a new notion of "random truth-table (and Turing) reducibility." This section will follow the idea of Ben-David et al. and introduce *probabilistic truth-table reducibility* among distributional decision problems. (This name seems more appropriate than "random reducibility" because of its similarity to worst-case probabilistic Turing reductions.)

5.5.1 Skew Bounded-Error Probabilistic Reducibility

Ben-David, Chor, Goldreich, and Luby [9] introduced a notion of *random reducibility* for which reduction machines are probabilistic rather than deterministic. Here we use the phrase *skew bounded-error probabilistic truth-table reducibility* to describe this type of reductions.

Definition 5.5.1 (Skew Bounded-Error Probabilistic Truth-Table Reductions) [9] Let (A, μ) and (B, ν) be any distributional decision problems.

1. (A, μ) is *skew bounded-error probabilistic polynomial-time truth-table reducible* (skew bpp-tt-reducible, for short) to (B, ν) if there exist a randomized oracle Turing machine M, a positive real number ϵ, and a semi-distribution η such that
   (i) (Query Type) M makes nonadaptive queries;
   (ii) (Efficiency) M with oracle B is polynomial-time bounded;
   (iii) (Validity) $\mathbf{Pr}_M[M^B(x) = A(x)] \geq \frac{1}{2} + \epsilon$ for all x; and
   (iv) (Domination) $\mu \preceq^{\mathrm{rp}}_{\Gamma_{M^B}} \eta$ and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{(x,s) \mid z \in Q(M,B,x,s)\})$.
2. (A, μ) is *skew bounded-error probabilistic average polynomial-time truth-table reducible* (skew avbpp-tt-reducible, for short) to (B, ν) if there exist a randomized oracle Turing machine M, a positive real number ϵ, and a semi-distribution η such that
   (i) (Query Type) M makes nonadaptive queries;
   (ii) (Efficiency) M with oracle B is polynomial-time bounded on μ-average;
   (iii) (Validity) $\mathbf{Pr}_M[M^B(x) = A(x)] \geq \frac{1}{2} + \epsilon$ for all x; and
   (iv) (Domination) $\mu \preceq^{\mathrm{avrp}}_{\Gamma_{M^B}} \eta$ and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{(x,s) \mid z \in Q(M,B,x,s)\})$.

Condition (iii) is also called the *domination condition for the reduction* M.

Ben-David, Chor, Goldreich, and Luby [9] instead used the following domination condition:

$$\hat{\mu}_{\Gamma_{M^B}}(\{(x,s) \mid z \in Q(M,B,x,s)\}) \leq p(|z|) \cdot \hat{\nu}(z)$$

for some polynomial p. Our condition (iv) is obviously weaker.

In worst-case complexity theory, we often assume that an oracle Turing machine queries only strings whose length is larger than that of the input. The motivation is that we can always construct another oracle machine which satisfies this condition without changing the complexity of computations, and also we can find another oracle set which is very close to the original oracle set. For example, assume that polynomial-time oracle Turing machine M with oracle set B computes a set A. We let $B' = \{\langle z, x\rangle \mid x \in B\}$ and let M' simulate M with the following change to oracle queries: if M queries a string z, then M' queries a pair $\langle z, x\rangle$. Then, clearly A is computed by M' with oracle B' and also $B' \leq^{\mathrm{p}}_m B$ holds.

We can obtain a result similar to one in worst-case complexity theory; however, its proof is more involved.

Lemma 5.5.2 *Assume that (A,μ) is skew avbpp-tt-reducible to (B,ν) with $B \neq \Sigma^*$. There exists a bounded-error probabilistic oracle Turing machine M' and a distributional problem (B',ν') such that*

(i) $(B',\nu') \leq^{\mathrm{p}}_m (B,\nu)$;

(ii) (A,μ) is skew avbpp-tt-reducible to (B',ν') via M'; and

(iii) all strings queried by M' with oracle B' on input x are of length greater than $|x|$.

Proof. Let us assume that (A,μ) is skew avbpp-tt-reducible to (B,ν) via a bounded-error probabilistic oracle Turing machine M with nonadaptive queries which runs in time p on μ-average, where p is an increasing polynomial. For simplicity, let Γ be the random-input domain associated with M^B. There exist a semi-distribution η and a random function q which is polynomial on μ-average such that $q(x,s)\cdot\hat{\eta}(x,s) \geq \hat{\mu}_\Gamma(x,s)$ and $\hat{\nu}(z) \geq \hat{\eta}(\{(x,s) \mid z \in Q(M,B,x,s)\})$ for all x and z. We can assume without loss of generality that $\mathrm{Time}^B_M(x) > 0$ for all strings x, and that on the empty input λ, M does not make any queries.

For brevity, we write $\delta(z,n) = \hat{\eta}(\{(x,s) \mid x \in \Sigma^n \wedge z \in Q(M,B,x,s)\})$ and $\delta(z) = \sum_{n=0}^{\infty}\delta(z,n)$. Notice that the sum $\sum_z \delta(z)$ does not exceed 1 because

$$\sum_z \delta(z) = \sum_z \hat{\eta}(\{(x,s) \mid z \in Q(M,B,x,s)\}) \leq \sum_z \hat{\nu}(z) = 1.$$

Moreover, we write $Q(M,B) = \bigcup_{(x,s)\in\Gamma} Q(M,B,x,s)$. To obtain the desired results, we let $B' = \{z01^n \mid z \in B, n \in \mathbb{N}\}$ and define ν' as follows:

$$\hat{\nu}'(w) = \begin{cases} \hat{\nu}(z)\cdot\frac{\delta(z,n)}{\delta(z)} & \text{if } w = z01^n \text{ and } \delta(z) > 0 \text{ for some } n \in \mathbb{N} \\ & \qquad \text{and some } z \in Q(M,B), \\ 0 & \text{otherwise.} \end{cases}$$

Since $\sum_{n=0}^{\infty}\hat{\nu}'(z01^n) = \hat{\nu}(z)$, we have

$$\sum_w \hat{\nu}'(w) = \sum_z\sum_{n=0}^{\infty}\hat{\nu}(z01^n) = \sum_z \hat{\nu}(z) = 1.$$

Thus, ν' becomes a distribution.

First we shall show that $(B',\nu') \leq^{\mathrm{p}}_m (B,\nu)$. Fix an element z_0 in $\overline{B}$ (because of $B \neq \Sigma^*$). Define a function f as follows:

$$f(w) = \begin{cases} z & \text{if } w = z01^n, \\ z_0 & \text{otherwise.} \end{cases}$$

It is clear that f is $\mathbf{P}$-computable, and f reduces B' to B. (Note that this f is not p-honest.) For the string z_0, $\hat{\nu}'_{f^{-1}}(z_0) = \hat{\nu}'(\{w \mid f(w) = z_0\}) = 0$ by the definition of f, and thus we have $\hat{\nu}(z_0) \geq \hat{\nu}'_{f^{-1}}(z_0)$. For other strings z,

$$\hat{\nu}(z) = \sum_{n=0}^{\infty}\hat{\nu}'(z01^n) = \hat{\nu}'(\{w \mid \exists z\exists n[w = z01^n]\}) = \hat{\nu}'(\{w \mid f(w) = z\}) = \hat{\nu}'_{f^{-1}}(z).$$

Hence, $(B',\nu') \leq^{\mathrm{p}}_m (B,\nu)$ via f.

We next show (ii) and (iii) of the lemma. We define a new randomized oracle Turing machine M' which works as follows: on input x, M' simulates M on input x, and whenever M queries a string z, M' queries $z01^{|x|}$ to the oracle. This simulation is carried out on each computation path. Note that the random-input domain associated with (M', B') is the same as that with (M, B). By definition, $z01^{|x|} \in Q(M', B', x, s)$ if and only if $z \in Q(M, B, x, s)$. Clearly M' with oracle B' computes A with bounded-error probability. Moreover, the length of a queried string is longer than that of the input. The rest of the proof will be devoted to showing the domination condition for (M', B').

Now we introduce a random function g. Define

$$g(x,s) = \begin{cases} \min\left\{ \frac{\delta(z,|x|)}{\delta(z)} \,\middle|\, \delta(z,|x|) > 0 \wedge z \in Q(M,B,x,s) \right\} & \text{if such a } z \text{ exists,} \\ 1 & \text{otherwise.} \end{cases}$$

Note that $0 < g(x,s) \leq 1$ for all x. In particular, $g(\lambda, s) = 1$ because M does not have any query strings. Using this function g, we also define η' as follows:

$$\hat{\eta}'(x,s) = \hat{\eta}(x,s) \cdot g(x,s).$$

We shall show that $\hat{\nu}'(z01^n) \geq \hat{\eta}'(\{(x,s) \mid z01^n \in Q(M', B', x, s)\})$.

Claim 13 $\hat{\nu}'(z01^n) \geq \hat{\eta}'(\{(x,s) \mid z01^n \in Q(M', B', x, s)\})$.

Proof of Claim. For every z in $Q(M, B, x, s)$, it follows that $\hat{\eta}'(x,s) \leq \hat{\eta}(x,s) \cdot \frac{\delta(z,n)}{\delta(z)}$. Hence, for every z and n, we have

$$\begin{aligned} &\hat{\eta}'(\{(x,s) \mid z01^n \in Q(M', B', x, s)\}) \\ &\quad = \hat{\eta}'(\{(x,s) \mid x \in \Sigma^n \wedge z \in Q(M, B, x, s)\}) \\ &\quad \leq \hat{\eta}(\{(x,s) \mid z \in Q(M, B, x, s)\}) \cdot \frac{\delta(z,n)}{\delta(z)} \\ &\quad \leq \hat{\nu}(z) \cdot \frac{\delta(z,n)}{\delta(z)} \quad = \quad \hat{\nu}'(z01^n). \end{aligned}$$

■

We next show that $\mu' \preceq^{\mathrm{avrp}}_{\Gamma} \eta'$. By the definition of η', it follows that

$$\hat{\mu}(x) \leq q(x,s) \cdot \hat{\eta}(x,s) = \frac{q(x,s)}{g(x,s)} \cdot g(x,s) \cdot \hat{\eta}(x) = \frac{q(x,s)}{g(x,s)} \cdot \hat{\eta}'(x,s).$$

To show the desired result, we should show that the random function $1/g$ is polynomial on μ-average since, if so, Lemma 3.3.13 ensures that the function q/g is also polynomial on μ-average. Let us assume that q is p' on μ-average, where p' is an increasing polynomial. For simplicity, write $Q_{x,s}$ for $Q(M, B, x, s)$ and let $\tilde{p}(z) = 6z^2 \cdot p'(3z)$. We shall show that $1/g$ is $\tilde{p}$ on μ-average. Assume $r \geq 1$. Let $E_r = \{(x,s) \in \Gamma \mid x \in \Sigma^+ \wedge \mathrm{Time}^B_M(x;s) \leq p(|x| \cdot 3r)\}$. Note that, for every $(x,s) \in E_r$, if $\hat{\mu}(x) > 0$, then $|s| \leq \mathrm{Time}^B_M(x;s) \leq p(|x| \cdot 3r)$. Moreover, in this case, we have $|\langle x, s\rangle| \leq 2|x| + |s| + 1 \leq 2|x|(p(|x| \cdot 3r) + 2)$.

This is seen as follows:

$$\begin{aligned}
&\hat{\mu}_\Gamma\left(\left\{x \;\middle|\; \frac{1}{g(x,s)} > \tilde{p}(|x|\cdot r)\right\}\right)\\
&\quad= \hat{\mu}_\Gamma(\{(x,s) \mid g(x,s)\cdot\tilde{p}(|x|\cdot r) < 1\})\\
&\quad\le \hat{\mu}_\Gamma(\{(x,s) \mid \mathrm{Time}_M^B(x;s) > p(|x|\cdot 3r)\}) + \hat{\mu}_\Gamma(\{(x,s) \mid q(x,s) > p'(|x|\cdot 3r)\})\\
&\qquad\quad +\hat{\mu}_\Gamma(\{(x,s)\in E_r \mid g(x,s)\cdot\tilde{p}(|x|\cdot r) < 1\}).
\end{aligned}$$

It is clear that the first two terms are bounded above by $1/3r$. Let T_r be the third term, and we shall show this term is also bounded by $1/3r$. It holds that

$$T_r \le \sum_{n=1}^{\infty} \hat{\mu}_\Gamma(\{(x,s)\in E_r \mid g(x,s)\cdot 6n^2 r\cdot p'(3nr) < 1\}).$$

Notice that $g(x,s) < 1$, and thus, $\delta(z,|x|) > 0$ for some $z \in Q_{x,s}$. Then, $g(x,s) = \frac{\delta(z,|x|)}{\delta(z)}$ for some z.

$$\begin{aligned}
T_r &\le \hat{\mu}_\Gamma(\{(x,s)\in E_r \mid g(x,s)\cdot 6|x|^2 r\cdot p'(|x|\cdot 3r) < 1\})\\
&\le \hat{\mu}_\Gamma(\{(x,s)\in E_r \mid \exists z\in Q_{x,s}[\delta(z,|x|)\cdot 6|x|^2 r\cdot p'(|x|\cdot 3r) < \delta(z)]\})\\
&\le \sum_{n=1}^{\infty} p'(3nr)\cdot\hat{\eta}\left(\left\{(x,s)\in E_r \;\middle|\; x\in\Sigma^n \wedge \exists z\in Q_{x,s}\left[\delta(z,n) < \frac{\delta(z)}{6rn^2\cdot p'(3nr)}\right]\right\}\right)\\
&\le \sum_{n=1}^{\infty}\sum_{z} p'(3nr)\cdot\hat{\eta}\left(\left\{(x,s) \;\middle|\; x\in\Sigma^n \wedge z\in Q_{x,y} \wedge \delta(z,n) < \frac{\delta(z)}{6rn^2\cdot p'(3nr)}\right\}\right)\\
&\le \sum_{n=1}^{\infty}\sum_{z} p'(3nr)\cdot\frac{\delta(z)}{6rn^2\cdot p'(3nr)} \;\le\; \sum_{n=1}^{\infty}\frac{1}{6rn^2}\\
&\le \frac{\pi^2}{36r} \;<\; \frac{1}{3r}.
\end{aligned}$$

□

5.5.2 More Structural Properties

We shall show another important lemma below. Before stating the lemma, we prepare some notation.

Let M be a randomized oracle Turing machine, and let N be a randomized Turing machine. We define another randomized Turing machine M_N which, on input x, does the following: it simulates M on input x; whenever M queries z, simulates N on input z; and halts if M does. Let $\tilde{\Gamma}$ be the the random-input domain of this composite machine M_N. Let x be fixed. For each random seed $r \in \tilde{\Gamma}(x)$, r_x denotes the associated random seed generated by M with some oracle on input x, and r_z denotes the associated random seed generated by N on input z.

Using this notation, the lemma is stated as follows.

Lemma 5.5.3 *Let μ and ν be distributions and let A be a set. Let M be a randomized oracle Turing machine and let N be a randomized Turing machine. Let g be a random function with random-input domain*

Γ *which is almost total. Assume that there exists a semi-distribution* η *satisfying the following condition:* $\mu \preceq^{\mathrm{avrp}}_{\Gamma_{M^A}} \eta$ *and* $\hat{\nu} \geq \lambda z.\hat{\eta}(\{(x,s) \mid z \in Q(M,A,x,s)\})$. *Define the random function* h *from* $\tilde{\Gamma}$ *to* $\mathbb{N}$ *as* $h(x,r) = \sum_{z \in Q(M,A,x,r_x)} g(z,r_z)$. *If* g *is polynomial on* ν*-average, then* h *is polynomial on* μ*-average.*

Proof. For brevity, we write Γ' in place of Γ_{M^A}. We choose an increasing polynomial p_A such that g is p_A on ν-average. Since $\mu \preceq^{\mathrm{avrp}}_{\Gamma'} \eta$, take an increasing polynomial p and a random function q such that $q(x,s) \cdot \hat{\eta}(x,s) \geq \hat{\mu}_{\Gamma'}(x,s)$ for all pairs $(x,s) \in \Gamma'$, and q is p on μ-average.

Set

$$\tilde{p}(z) = p_A(3z) \cdot p_A(p_A(3z) \cdot 6z^2 p(3z)) + c_0,$$

where $c_0 = \max_{r \in \tilde{\Gamma}(x)} h(\lambda, r)$. We shall show that h is $\tilde{p}$ on μ-average. Fix $d \geq 1$. Let E_d and $\tilde{E}_d$ be the sets defined by

$$\begin{aligned} E_d &= \{(x,s) \in \Gamma' \mid x \in \Sigma^+ \wedge q(x,s) \leq p(|x| \cdot 3d) \wedge \mathrm{Time}^A_M(x;s) \leq p_A(|x| \cdot 3d)\}; \text{ and} \\ \tilde{E}_d &= \{(x,r) \in \tilde{\Gamma} \mid (x,s_x) \in E_d\}. \end{aligned}$$

Let $Q_{x,s}$ denote $Q(M,A,x,s)$. We estimate the term $\hat{\mu}_{\tilde{\Gamma}}(\{(x,r) \mid h(x,r) > \tilde{p}(|x| \cdot r)\})$ as follows:

$$\begin{aligned} &\hat{\mu}_{\tilde{\Gamma}}(\{(x,r) \mid h(x,r) > \tilde{p}(|x| \cdot d)\}) \\ &\quad \leq \hat{\mu}_{\tilde{\Gamma}}(\{(x,r) \mid q(x,r_x) > p(|x| \cdot 3d)\}) + \hat{\mu}_{\tilde{\Gamma}}(\{(x,r) \mid \mathrm{Time}^A_M(x;r_x) > p_A(|x| \cdot 3d)\})\}) \\ &\qquad + \hat{\mu}_{\tilde{\Gamma}}\left(\left\{(x,r) \in \tilde{E}_d \;\middle|\; \sum_{z \in Q_{x,r_x}} g(z,r_z) > \tilde{p}(|x| \cdot d)\right\}\right). \end{aligned}$$

Clearly the first term is

$$\hat{\mu}_{\tilde{\Gamma}}(\{(x,r) \mid q(x,r_x) > p(|x| \cdot 3d)\}) = \hat{\mu}_{\Gamma'}(\{(x,s) \mid q(x,s) > p(|x| \cdot 3d)\}) < \frac{1}{3d}.$$

Similarly, the second term is also bounded above by $1/3d$. Let us denote the last term by T_d and calculate its upper bound.

Fix $(x,r) \in \tilde{E}_d$ and let $|x| = n$ $(n > 0)$. Assume that $\sum_{z \in Q_{x,r_x}} g(z,r_z) > \tilde{p}(dn)$. Then, for some $z \in Q_{x,r_x}$ we get $\|Q_{x,r_x}\| \cdot g(z,r_z) > \tilde{p}(dn)$. Fix such a string z. Since $\|Q_{x,r_x}\| \leq \mathrm{Time}^A_M(x;r_x) \leq p_A(3dn)$, we have $g(z,r_z) > p_A(p_A(3dn) \cdot 6n^2 d \cdot p(3dn))$. Moreover, $|z| \leq \mathrm{Time}^A_M(x;r_x) \leq p_A(3dn)$. Thus, it follows that $g(z,r_z) > p_A(|z| \cdot 6n^2 d \cdot p(3dn))$. Using this fact, the term T_d is estimated as follows:

$$\begin{aligned} T_d &\leq \sum_{n=1}^{\infty} \hat{\mu}_{\tilde{\Gamma}}(\{(x,r) \in \tilde{E}_d \mid x \in \Sigma^n \wedge \exists z \in Q_{x,r_x}[g(z,r_z) > p_A(|z| \cdot 6dn^2 \cdot p(3dn))]\}) \\ &\leq \sum_{n=1}^{\infty} \widehat{\mu_{\Gamma'} \times \gamma}(\{(x,s,s') \mid (x,s) \in E_d \wedge x \in \Sigma^n \\ &\qquad \wedge \exists z \in Q_{x,s}[s' \in \Gamma(z) \wedge g(z,s') > p_A(|z| \cdot 6dn^2 \cdot p(3dn))]\}), \end{aligned}$$

where $\widehat{\mu_{\Gamma'} \times \gamma}(x,s,s') = \hat{\mu}_{\Gamma'} \cdot 2^{-|s'|}$.

Write $\eta \times \gamma$ to denote the distribution ξ defined as $\hat{\xi}(x,s,s') = \hat{\eta}(x,s) \cdot 2^{-|s'|}$ for all (x,s,s'). From the fact that $p(3dn) \cdot \hat{\eta}(x,s) \geq \hat{\mu}_{\Gamma_{M'}}(x,s)$ for each $(x,s) \in E_d$, it follows that $p(3dn) \cdot \widehat{\eta \times \gamma}(x,s,s') \geq \widehat{\mu_{\Gamma'} \times \gamma}(x,s,s')$.

Thus, T_r is bounded above by the term

$$\sum_{n=1}^{\infty} p(3dn) \cdot \widehat{\eta \times \gamma}(\{(x,s,s') \mid (x,s) \in E_d \wedge x \in \Sigma^n \\ \wedge \exists z \in Q_{x,s}[s' \in \Gamma(z) \wedge g(z,s') > p_A(|z| \cdot 6dn^2 \cdot p(3dn))]\}).$$

Since $\hat{\nu}(B) \geq \hat{\eta}(\{(x,s) \mid \exists z \in Q_{x,s}[z \in B]\})$ for any set B,

$$\hat{\nu}_\Gamma(\{(z,s') \mid g(z,s') > p_A(|z| \cdot p(3dn) \cdot 6dn^2)\}) \\ \geq \widehat{\eta \times \gamma}(\{(x,s,s') \mid \exists z \in Q_{x,s}[g(z,s') > p_A(|z| \cdot p(3dn) \cdot 6dn^2)]\}).$$

As a conclusion,

$$\begin{aligned} T_d &\leq \sum_{n=1}^{\infty} p(3dn) \cdot \hat{\nu}_\Gamma(\{(z,s') \mid g(z,s') > p_A(|z| \cdot p(3dn) \cdot 6dn^2)\}) \\ &\leq \sum_{n=1}^{\infty} \frac{p(3dn)}{p(3dn) \cdot 6dn^2} = \frac{\pi^2}{36d} < \frac{1}{3d}. \end{aligned}$$

□

Skew avbpp-tt-reducibility turns out to be closed under p-m-reducibility.

Lemma 5.5.4 *Assume that $(A,\mu) \leq_m^{\mathrm{p}} (B,\nu)$ and (B,ν) is skew avbpp-tt-reducible to (C,ξ). Then, (A,μ) is skew avbpp-tt-reducible to (C,ξ).*

Proof. For simplicity, let us assume that $(A_1,\mu_1) \leq_m^{\mathrm{p}} (A_2,\mu_2)$ and that (A_2,μ_2) is skew avbpp-tt-reducible to (A_3,μ_3). Let f be an appropriate reduction which reduces (A_1,μ_1) to (A_2,μ_2) with the domination condition for f. Further, let M be a bounded-error probabilistic oracle Turing reduction which reduces (A_2,μ_2) to (A_3,μ_3).

We shall consider the randomized algorithm N defined as follows: on input x, first compute $f(x)$ and then simulate M on input $f(x)$. For each fixed x, since $A_1(x) = A_2(f(x))$,

$$\mathbf{Pr}_N[N^{A_3}(x) = A_1(x)] = \mathbf{Pr}_M[M^{A_3}(f(x)) = A_2(f(x))] \geq \frac{2}{3}.$$

This shows that N is a bounded-error probabilistic oracle Turing machine with oracle A_3.

The estimation of the running time of N with oracle A_3 on input x with random input s, $\mathrm{Time}_N^{A_3}(x;s)$, is given by

$$\mathrm{Time}_N^{A_3}(x;s) \leq c \cdot (\mathrm{Time}_f(x) + \mathrm{Time}_M^{A_3}(f(x)) + 1)$$

for some constant $c > 0$. Notice that $\lambda x.\mathrm{Time}_f(x)$ is polynomial on μ_1-average, and $\lambda xs.\mathrm{Time}_M^{A_3}(f(x))$, as a random function, is also polynomial on μ_1-average by Lemma 5.2.4. In consequence, $\lambda xs.\mathrm{Time}_N^{A_3}(x;s)$ is polynomial on μ_1-average.

To see the domination condition for N, write η_2' for the default distribution induced from η_2 and $\Gamma_{N^{A_3}}$. Then,

$$\hat{\mu}_3(z) \geq \hat{\eta}_2'(\{(w,s) \mid z \in Q(M,A_3,w,s)\}) \geq \sum_{(w,s) \in \Gamma_{M^{A_3}}} \hat{\eta}_2(w) \cdot 2^{-|s|} \geq \sum_{(w,s) \in \Gamma_{M^{A_3}}} \frac{\hat{\mu}_2(w)}{p_M(w,s)} \cdot 2^{-|s|}.$$

For w, it holds that

$$\frac{\hat{\mu}_2(w)}{p_M(w,s)} \geq \frac{\hat{\eta}_1(f^{-1}(w))}{p_M(w,s)} \geq \sum_{x \in f^{-1}(w)} \frac{\hat{\mu}_1(x)}{p_M(w,s) \cdot p_f(x)} = \sum_{x \in f^{-1}(w)} \frac{\hat{\mu}_1(x)}{p_M(f(x),s) \cdot p_f(x)}.$$

Thus,

$$\hat{\mu}_3(z) \geq \sum_{(w,s) \in \Gamma_{M^{A_3}}} \sum_{x \in f^{-1}(w)} \frac{\hat{\mu}_1(x)}{p_M(f(x),s) p_f(x)} \cdot 2^{-|s|}.$$

This sum is taken over all pairs $(x,s) \in \Gamma_{N^{A_3}}$ such that

$$\exists w \in \mathrm{ran}(f)[f(x) = w \wedge z \in Q(M, A_3, w, s)].$$

This condition is equivalent to the condition $z \in Q(N, A_3, x, s)$. Now let us define $p(z,s) = p_M(f(x),s) \cdot p_f(z)$ and $\hat{\eta}(x,s) = \hat{\mu}_1(x) \cdot 2^{-|s|}/p(x,s)$ if $s \in \Gamma_{N^{A_3}}(x)$, and 0 otherwise. Notice that $\lambda x.|f(x)|$ is polynomial on μ_1-average, and consequently, $\lambda x s.p_M(f(x),s)$ is polynomial on μ_1-average. Thus, p becomes polynomial on μ_1-average. Then it follows that

$$\hat{\mu}_3(z) \geq \sum_{(x,s) \in \Gamma_{N^{A_3}}} \frac{\hat{\mu}_1(x)}{p(x,s)} = \sum_{(x,s) \in \Gamma_{N^{A_3}}} \hat{\eta}(x,s) = \hat{\eta}(\{(x,s) \mid z \in Q(N, A_3, x, s)\}).$$

This completes the proof. □

We need a relativized version of the Amplification Lemma (Lemma 3.5.31).

Lemma 5.5.5 *Assume that (A,μ) is skew avbpp-tt-reducible to (B,ν) with $B \neq \Sigma^*$. Then, there exist a semi-distribution η', a randomized Turing machine N, and a distributional decision problem (B',ν') such that*

(i) $(B',\nu') \leq_m^{\mathrm{p}} (B,\nu)$;

(ii) $\mathbf{Pr}_s[N^{B'}(x,1^m;s) = A(x) \mid s \in \Gamma_{N^{B'}}(x,1^m)] \geq 1 - 2^{-|x|-m}$ *for all* $x \in \Sigma^*$ *and* $m \in \mathbb{N}$; *and*

(iii) $\mu \times \nu_{\mathrm{tally}} \preceq^{\mathrm{avrp}}_{\Gamma_{N^{B'}}} \eta'$ *and* $\hat{\nu}' \geq \lambda z.\hat{\eta}'(\{(x,y,s) \mid z \in Q(N, B', x, y, s)\})$.

Proof. Let us assume that (A,μ) is skew avbpp-tt-reducible to (B,ν) via a randomized Turing machine M and a semi-distribution η. Let Γ be the random-input domain associated with M with oracle B. We suppose that $\mathbf{Pr}_M[M^B(x) = A(x)] \geq \frac{1}{2} + \epsilon$. Since $\mu \preceq^{\mathrm{avrp}}_{\Gamma} \eta$, there is a random function q which is polynomial on μ-average such that $q(x,s) \cdot \hat{\eta}(x,s) = \hat{\mu}_\Gamma(x,s)$ for all pairs $(x,s) \in \Gamma$. Let z_0 be the minimal string that is not in B. We can assume that M does not query any strings of length smaller than or equal to $|z_0|$.

For the desired problem (B',ν'), we set $B' = \{z01^n \mid z \in B \wedge n \in \mathbb{N}\}$ and set

$$\hat{\nu}'(w) = \begin{cases} \hat{\nu}_{\mathrm{tally}}(1^n) \cdot \hat{\nu}(z) & \text{if } w = z01^n \text{ for some } n \in \mathbb{N}, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously ν' becomes a distribution because $\sum_{n=1}^{\infty} \hat{\nu}'(z01^n) = \hat{\nu}(z)$ for all strings z.

We then show (i). Let f be

$$f(w) = \begin{cases} z & \text{if } w = z01^n \text{ for some } n \in \mathbb{N}, \\ z_0 & \text{otherwise.} \end{cases}$$

The function f becomes a reduction function which reduces B' to B. To show that $(B', \nu') \leq_m^{\mathrm{p}} (B, \nu)$, we should check if the domination condition for f holds. This is shown as follows:

$$\hat{\nu}'(f^{-1}(z)) = \sum_{n=1}^{\infty} \hat{\nu}'(z01^n) = \sum_{n=1}^{\infty} \hat{\nu}_{\mathrm{tally}}(1^n) \cdot \hat{\nu}(z) = \hat{\nu}(z) \cdot \hat{\nu}_{\mathrm{tally}}(\{1\}^*) = \hat{\nu}(z).$$

Next we show (ii). Let us recall the randomized algorithm in the proof of the Amplification Lemma that boosts the success probability up to $1 - 2^{-|x|-m}$ on input pair $(x, 1^m)$. Here we slightly modify its algorithm to allow the algorithm to make queries. Take an integer c satisfying $c > 1/\epsilon$ and let $p(x, y) = 2c^3(|x| + |y|)$ for all pairs (x, y). Then, we let N be the randomized oracle Turing machine defined by the following algorithm:

begin randomized algorithm for N
  input (x, y)
  if $y \notin \{1\}^*$ **then reject**
  for $i = 1$ **to** $p(x, y)$ **do**
    simulate M on input x without queries until M produces a query list
  end for
  list all possible query strings
  for $i = 1$ **to** $p(x, y)$ **do**
    while simulation **do**
      if M makes a query z **then** query $z01^i$
    end while
  end for
  if the majority of the outcomes is 1 **then accept else reject**
end.

By a similar argument to that in the proof of the Amplification Lemma, we can prove (ii).

Notice that, for any strings x and y and any random seed s, the relation $z01^n \in Q(N, B', x, y, s)$ implies that $y = 1^m$ and $1 \leq n \leq p(x, 1^m)$ for some number $m \in \mathbb{N}$. Moreover, provided that $1 \leq n \leq p(x, 1^m)$, we have the close relationship

$$z01^n \in Q(N, B', x, 1^m, s) \Longleftrightarrow z \in Q(M, B, x, s_n),$$

where each s_i is in $\Gamma(x)$ and is associated with s.

We shall show (iii). Let Γ' be the random-input domain associated with N with oracle B'. Then, we define η' as follows. For $(x, y, s) \in \Gamma'$, let us define $q'(x, y, s)$ to be $8(p(x, y) + 1)^2 \cdot \sum_{i=1}^{p(x,y)} q(x, s_k)$ and let

$$\hat{\eta}'(x, y, s) = \hat{\mu}(x) \cdot \hat{\nu}_{\mathrm{tally}}(y) \cdot 2^{-|s|} \cdot \frac{1}{q'(x, y, s)}.$$

It is not difficult to see that q' is polynomial on $\mu \times \nu_{\rm tally}$-average and also $\mu \times \nu_{\rm tally} \preceq^{\rm avrp}_{\Gamma'} \eta'$.

By the definition of η', it follows that, for any $(x,y,s) \in \Gamma'$,

$$\begin{aligned}
\hat{\eta}'(x,y,s) &= \frac{\hat{\mu}(x)\cdot\hat{\nu}_{\rm tally}(y)\cdot 2^{-|s|}}{q'(x,y,s)} = \frac{\hat{\mu}(x)\cdot\hat{\nu}_{\rm tally}(y)\cdot 2^{-|s|}}{8(p(x,y)+1)^2\sum_{i=1}^{p(x,y)} q(x,s_i)} \\
&\le \hat{\nu}_{\rm tally}(1^{p(x,y)})\cdot\hat{\nu}_{\rm tally}(y)\cdot\sum_{i=1}^{p(x,y)}\frac{\hat{\mu}_\Gamma(x)}{q(x,s_i)} \\
&= \hat{\nu}_{\rm tally}(1^{p(x,y)})\cdot\hat{\nu}_{\rm tally}(y)\cdot\sum_{i=1}^{p(x,y)}\hat{\eta}(x,s_i)\cdot 2^{-|s|+|s_i|}.
\end{aligned}$$

Now we must check the domination condition for N; that is, $\hat{\eta}'(\{(x,y,s) \mid z01^n \in Q(N,B',x,y,s)\}) \le \hat{\nu}'(z01^n)$ for any z and n. Using the previous calculation, for each string of the form $z01^m$,

$$\begin{aligned}
&\hat{\eta}'(\{(x,y,s) \mid z01^n \in Q(N,B',x,y,s)\}) \\
&\quad= \sum_{m=0}^{\infty}\sum_{(x,s)\in\Gamma'}\hat{\eta}'(x,1^m,s)\cdot[z01^n \in Q(N,B',x,1^m,s)] \\
&\quad\le \sum_{m=0}^{\infty}\sum_{(x,s)\in\Gamma'}\hat{\nu}_{\rm tally}(1^{p(x,1^m)})\cdot\hat{\nu}_{\rm tally}(1^m)\cdot\sum_{i=1}^{p(x,1^m)}\hat{\eta}(x,s_i)\cdot 2^{-|s|+|s_i|}\cdot[z01^n \in Q(N,B',x,1^m,s)].
\end{aligned}$$

The last term is equivalent to the following term:

$$\sum_{m=0}^{\infty}\sum_{(x,s_n)\in\Gamma}\hat{\nu}_{\rm tally}(1^n)\cdot\hat{\nu}_{\rm tally}(1^m)\cdot\hat{\eta}(x,s_n)\cdot[z \in Q(M,B,x,s_n)].$$

Therefore,

$$\begin{aligned}
&\hat{\eta}'(\{(x,y,s) \mid z01^n \in Q(N,B',x,y,s)\}) \\
&\quad\le \hat{\nu}_{\rm tally}(1^n)\cdot\sum_{m=0}^{\infty}\left(\sum_{(x,s')\in\Gamma}\hat{\eta}(x,s')\cdot[z \in Q(M,B,x,s')]\right)\cdot\hat{\nu}_{\rm tally}(1^m) \\
&\quad\le \hat{\nu}_{\rm tally}(1^n)\cdot\hat{\eta}'(\{(x,s') \mid z \in Q(M,B,x,s')\})\cdot\sum_{m=0}^{\infty}\hat{\nu}_{\rm tally}(1^m) \\
&\quad\le \hat{\nu}_{\rm tally}(1^n)\cdot\hat{\nu}(z) = \hat{\nu}'(z01^n).
\end{aligned}$$

This completes the proof. □

5.5.3 Bounded Error Probabilistic Truth Table Reducibility

An ordinary bounded-error probabilistic oracle Turing machine can diminish the error probability significantly by repeating the same computation and taking a majority vote to determine the outcomes of the machine. Our skew reduction does not seem to enjoy this property because of tight domination conditions for the reduction machines. To guarantee it, we need a many-one transformation as well as the skew reduction.

Now we introduce our bounded-error probabilistic truth-table reducibility.

Definition 5.5.6 (Bounded-Error Probabilistic Truth-Table Reductions) Let (A,μ) and (B,ν) be distributional decision problems.

1. (A,μ) is *bounded-error probabilistic polynomial-time truth-table reducible* (bpp-tt-reducible, for short) to (B,ν), denoted by $(A,\mu) \leq_{tt}^{\mathrm{bpp}} (B,\nu)$, if there exists a distributional decision problem (B',ν') such that (A,μ) is skew bpp-tt-reducible to (B',ν'), and (B',ν') is p-m-reducible to (B,ν).

2. (A,μ) is *bounded-error probabilistic average polynomial-time truth-table reducible* (avbpp-tt-reducible, for short) to (B,ν), denoted by $(A,\mu) \leq_{tt}^{\mathrm{avbpp}} (B,\nu)$, if there exist a distributional decision problem (B',ν') such that (A,μ) is skew avbpp-tt-reducible to (B',ν'), and (B',ν') is p-m-reducible to (B,ν).

In the following, we show basic properties of the reducibilities.

Proposition 5.5.7 *Let $\alpha \in \{\mathrm{bpp}, \mathrm{avbpp}\}$.*

1. *The relation $\leq_m^{\mathrm{p}}$ implies $\leq_{tt}^{\mathrm{bpp}}$, and $\leq_m^{\mathrm{avp}}$ implies $\leq_{tt}^{\mathrm{avbpp}}$.*

2. *The relation $\leq_{tt}^{\alpha}$ is reflexive.*

3. *The relation $\leq_{tt}^{\mathrm{bpp}}$ implies $\leq_{tt}^{\mathrm{avbpp}}$.*

4. *The relations $\leq_{tt}^{\mathrm{bpp}}$ and $\leq_{tt}^{\mathrm{avbpp}}$ are transitive.*

Proof. (1) By definition, deterministic Turing machines are a special case of probabilistic Turing machines.

(2) The claim of reflexivity is obvious by choosing the identity reduction, i.e., $f(x) = x$.

(3) Clear from the definitions.

(4) Here we show that $\leq_{tt}^{\mathrm{avbpp}}$ is transitive. Assuming that $(A_1,\mu_1) \leq_{tt}^{\mathrm{avbpp}} (A_2,\mu_2)$ and $(A_2,\mu_2) \leq_{tt}^{\mathrm{avbpp}} (A_3,\mu_3)$, we shall show that $(A_1,\mu_1) \leq_{tt}^{\mathrm{avbpp}} (A_3,\mu_3)$.

Let (A_2',μ_2') be a distributional problem such that (A_1,μ_1) is skew avbpp-tt-reducible to (A_2',μ_2') , and (A_2',μ_2') is p-m-reducible to (A_2,μ_2). Suppose that (A_2,μ_2) is skew avbpp-tt-reducible to some distributional problem, say (A_3',μ_3'), which is p-m-reducible to (A_3,μ_3). By Lemma 5.5.4, (A_2',μ_2') is skew avbpp-tt-reducible to (A_3',μ_3'). Apply Lemma 5.5.5 to this skew avbpp-tt-reduction to obtain another randomized oracle Turing machine N, another semi-distribution η, and another distributional problem (A_3'',μ_3'') which satisfies conditions (i)-(iii) of the lemma as well as $(A_3'',\mu_3'') \leq_m^{\mathrm{p}} (A_3,\mu_3)$.

For readability and simplicity, in what follows, we can assume that (A_1,μ_1) is skew avbpp-tt-reducible to (A_2,μ_2) via a semi-distribution η_1 and a bounded-error probabilistic oracle Turing machine M_1 which, with oracle A_2, is polynomial-time bounded on μ_1-average. Moreover, $\mathbf{Pr}_s[M_2^{A_3}(x,1^m;s) = A_2(x) \mid s \in \Gamma_{M_2^{A_3}}(x,1^m)] \geq 1 - 2^{-|x|-m}$, $\mu_2 \times \nu_{\mathrm{tally}} \preceq_{\Gamma_{M_2^{A_3}}}^{\mathrm{avrp}} \eta_2$, and $\hat{\mu}_3 \geq \lambda z.\hat{\eta}_2(\{(x,y,s) \mid z \in Q(M_2,A_3,x,y,s)\})$ for some semi-distribution η_2 and some randomized oracle Turing machine M_2 which, with oracle A_3, runs in polynomial time on μ_2-average.

By the domination conditions, there are random functions p_1 and p_2 which are polynomial on μ_1-average and polynomial on $\mu_2 \times \nu_{\mathrm{tally}}$-average, respectively, such that $p_1(x,s) \cdot \hat{\eta}_1(x,s) \geq \hat{\mu}_1(x) \cdot 2^{-|s|}$, $p_2(x,y,s) \cdot$

$\hat{\eta}_2(x,y,s) \geq \hat{\mu}_2(x) \cdot \hat{\nu}_{\mathrm{tally}}(y) \cdot 2^{-|s|}$, $\hat{\mu}_2(z) \geq \hat{\eta}_1(\{(x,s) \mid z \in Q(M_1, A_2, x, s)\})$, and $\hat{\mu}_3(z) \geq \hat{\eta}_2(\{(x,y,s) \mid z \in Q(M_2, A_3, x, y, s)\})$ for all x and z.

Now we define a randomized Turing machine N which computes A_1 using oracle A_3:

begin randomized algorithm for N
  input x (say, $n = |x|$)
  simulate M_1 on input x until the first query is made
  complete a list of query strings
  let m_{r_x} be the number of query strings
  resume the simulation
  while simulation **do**
    if M_2 queries z **then** simulate M_2 on input $(z, 1^{m_{r_x}+3})$
    if M_1 reaches a halting configuration **then** output $M_1(x)$ and **halt**
  end while
end.

Let r be the random seed generated by N with oracle A_3 on input x. Clearly random seed r is associated with all random seeds generated by M_2 with oracle A_2 on input strings which M_1 with oracle A_2 queries. For such random seeds, we write r_x to designate the random seed that is used for the computation of M_1 with oracle A_2 on input x, and write r_z for the random seed for the computation of M_2 with oracle A_3 on input z provided that z is in $Q(M_1, A_2, x, r_x)$.

Given a pair $(x,r) \in \Gamma_{N^{A_3}}$, the running time of N with A_3 as an oracle on input x with random input r is bounded by

$$c \cdot \left(\mathrm{Time}_{M_1}^{A_2}(x; r_x) + \sum_{z \in Q(M_1, A_2, x, r_x)} \mathrm{Time}_{M_2}^{A_3}(z, 1^{m_{r_x}+3}; r_z) + 1 \right)$$

for some constant $c > 0$, where m_{r_x} is the number of strings in the query list produced by N on input x along with random seed R_x. Clearly $\mathrm{Time}_{M_1}^{A_2}(x; r_x) \geq m_{r_x} \geq \|Q(M_1, A_2, r_x)\|$. It is immediate from Lemma 5.5.3 that the random function $\lambda x r. \sum_{z \in Q(M_1, A_2, x, r_x)} \mathrm{Time}_{M_2}^{A_3}(z, 1^{m_{r_x}+3}; r_z)$ is polynomial on μ_1-average; thus, N with oracle A_3 is polynomial-time bounded on μ_1-average.

To see the validity of this algorithm, let us consider the success probability of the new machine N. Fix an input x and a random seed r in $\Gamma_{N^{A_3}}(x)$. Write $Q_{x,s}$ for $Q(M_1, A_2, x, s)$. The success probability that $N^{A_3}(x; r) = A_2(x)$ is at least

$$\begin{aligned}
&\prod_{z \in Q(M_1, A_2, x, r_x)} \mathbf{Pr}_s[M_2^{A_3}(z, 1^{m_{r_x}+2}; s) = A_3(z) \mid s \in \Gamma_{M_2}^{A_3}(x, 1^{m_{r_x}+3})] \\
&\qquad \geq \prod_{z \in Q_{x,r_x}} (1 - 2^{-m_{r_x}-3}) \\
&\qquad \geq \prod_{z \in Q_{x,r_x}} \left(1 - 2^{-\|Q_{x,r_x}\|-2}\right) \\
&\qquad \geq \left(1 - 2^{-\|Q_{x,r_x}\|-3}\right)^{\|Q_{x,r_x}\|} \quad \geq \quad 1 - 2^{-\|Q_{x,r_x}\|-2+\|Q_{x,r_x}\|+1}
\end{aligned}$$

$$= \quad 1 - 2^{-2} \quad = \quad \frac{3}{4}.$$

It remains to establish the domination condition for N. For each z,

$$\hat{\mu}_3(z) \geq \sum_{(w,y,s')} \frac{\hat{\mu}_2(w) \cdot \hat{\nu}_{\mathrm{tally}}(y)}{p_2(w,y,s')} \cdot 2^{-|s|} \cdot [z \in Q(M_2, A_3, w, y, s')].$$

Write $p_2'(w,y,s')$ for $p_2(w,y,s') \cdot 8(|y|+1)^2$. Using the fact that $8(|y|+1)^2 \cdot \hat{\nu}_{\mathrm{tally}}(y) \geq 1$,

$$\hat{\mu}_3(z) \geq \sum_{(w,y,s')} \frac{\hat{\mu}_2(w)}{p_2'(w,y,s')} \cdot 2^{-|s|} \cdot [z \in Q(M_2, A_3, w, y, s')].$$

Also we have

$$\hat{\mu}_2(w) \geq \sum_{(x,s)} \frac{\hat{\mu}_1(x)}{p_1(x,s)} \cdot 2^{-|s|} \cdot [w \in Q(M_1, A_2, x, s)].$$

Combining both inequalities, we can calculate the lower bound of $\hat{\mu}_3(z)$ as follows:

$$\begin{aligned}
\hat{\mu}_3(z) &\geq \sum_{(w,y,s')} \sum_{(x,s)} \frac{\hat{\mu}_1(x)}{p_1(x,s) \cdot p_2'(w,y,s')} \cdot 2^{-|s|-|s'|}[w \in Q(M_1, A_2, x, s)] \cdot [z \in Q(M_2, A_3, w, y, s')] \\
&\geq \sum_{(x,r)} \frac{\hat{\mu}_1(x)}{p_1(x,r_x)} \cdot 2^{-|r_x|} \cdot \sum_{w \in Q_{x,r_x}} \frac{1}{p_2'(w, 1^{m_{r_x}+3}, r_w)} \cdot 2^{-|r_w|} \\
&= \sum_{(x,r)} \frac{\hat{\mu}_1(x)}{p_1(x,r_x)} \cdot 2^{-|r|} \cdot \sum_{w \in Q_{x,r_x}} \frac{1}{p_2'(w, 1^{m_{r_x}+3}, r_w)} \cdot 2^{|r|-|r_x|-|r_w|}.
\end{aligned}$$

Since $|r| \geq |r_x| + |r_w|$, we further calculate the lower bound as follows:

$$\begin{aligned}
\hat{\mu}_3(z) &\geq \sum_{(x,r)} \frac{\widehat{\mu_1 \times \gamma}(x,r)}{p_1(x,r_x)} \cdot \sum_{w \in Q_{x,r_x}} \frac{1}{p_2'(w, 1^{m_{r_x}+3}, r_w)} \\
&\geq \sum_{(x,r)} \frac{\widehat{\mu_1 \times \gamma}(x,r)}{p_1(x,r_x) \cdot \sum_{w \in Q_{x,r_x}} p_2'(w, 1^{m_{r_x}+3}, r_w)},
\end{aligned}$$

where $Q_{x,s} = Q(M_1, A_2, x, s)$ and $\widehat{\mu_1 \times \gamma}(x,r) = \hat{\mu}_1(x) \cdot 2^{-|r|}$ if $r \in \Gamma_{M_1^{A_2}}(x)$, and 0 otherwise.

Now we set $p(x,r) = p_1(x,r_x) \cdot \sum_{w \in Q(M_1, A_2, x, r_x)} p_2'(w, 1^{m_{r_x}+3}, r_w)$ and $\hat{\eta}(x,r) = \widehat{\mu_1 \times \gamma}(x,r)/p(x,r)$. By Lemma 5.5.3, p turns out to be polynomial on μ_1-average. It then follows that

$$\hat{\mu}_3(z) \geq \hat{\eta}(\{(x,r) \mid z \in Q(N, A_3, x, r)\}).$$

Thus, the claim is established. □

Finally we can show the closure property of the class Aver(**BPP**, $*$) under avbpp-reductions. This is a direct consequence of Proposition 5.5.7(4).

Theorem 5.5.8 **[9]** *The class* Aver(**BPP**, $*$) *is closed downward under avbpp-tt-reductions.*

Proof. Similar to that of Theorem 5.2.12. □

5.5.4 Application of Probabilistic Reducibility

The motivation in having introduced the bounded-error probabilistic reducibility is to show that distributional problems in Dist($\mathbf{NP}, \mathbf{P}$-comp) with natural flat distributions become complete. As a simple example, we shall demonstrate that a flat version of the randomized bounded halting problem is truly complete for Dist($\mathbf{NP}, \mathbf{P}$-comp) with respect to bpp-tt-reducibility.

Let us introduce a flat version of the *randomized bounded halting problem* (RBHP_{flat}), ($\mathrm{BHP}_{flat}, \mu_{\mathrm{BHP}_{flat}}$), in the following fashion:

$$\mathrm{BHP}_{flat} = \{\langle s_i, x, t\rangle \mid M_i \text{ accepts } x \text{ in } |t| \text{ time }\}; \text{ and}$$

$$\hat{\mu}_{\mathrm{BHP}_{flat}}(\langle s_i, x, t\rangle) = \hat{\nu}_{\mathrm{stand}}(s_i) \cdot \hat{\nu}_{\mathrm{stand}}(x) \cdot \hat{\nu}_{\mathrm{stand}}(t).$$

Theorem 5.5.9 RBHP_{flat} *is bpp-tt-complete for* Dist($\mathbf{NP}, \mathbf{P}$-comp).

Proof. It is enough for us to show that $(\mathrm{BHP}, \mu_{\mathrm{BHP}}) \leq_{tt}^{\mathrm{bpp}} (\mathrm{BHP}_{flat}, \mu_{\mathrm{BHP}_{flat}})$. Let us consider the following randomized Turing machine N:

begin randomized algorithm for N
  input $\langle s_i, x, 1^m\rangle$
  generate a random seed w of length m
  query $\langle s_i, x, w\rangle$ to oracle
  if $\langle s_i, x, w\rangle$ is in oracle **then accept else reject**
end.

Notice that there is no error if BHP_{flat} is chosen as an oracle; that is, $\mathbf{Pr}_s[N^{\mathrm{BHP}_{flat}}(\langle s_i, x, 1^m\rangle; s) = A(x)] = 1$. Also N makes nonadaptive queries and runs in polynomial time for any oracle.

Let c be an integer such that $c \geq 1/\hat{\nu}_{\mathrm{stand}}(s_i)$. Let us define the semi-distribution η as $\hat{\eta}(z, s) = \hat{\mu}_{\mathrm{BHP}}(z) \cdot 2^{-|s|}$ for all pairs $(z, s) \in \Gamma_{N^{\mathrm{BHP}}}$; otherwise, 0. We then have

$$\begin{aligned} \hat{\eta}(\{(z, s) \mid \langle s_i, x, w\rangle \in Q(N, \mathrm{BHP}_{flat}, z, s)\}) &= \hat{\eta}(\langle s_i, x, 1^{|w|}\rangle) \\ &= \hat{\mu}_{\mathrm{BHP}_{flat}}(\langle s_i, x, w\rangle). \end{aligned}$$

Therefore, we have $(\mathrm{BHP}, \mu_{\mathrm{BHP}}) \leq_{tt}^{\mathrm{bpp}} (\mathrm{BHP}_{flat}, \mu_{\mathrm{BHP}_{flat}})$. □

Another significant application of the bpp-tt-reducibility given by Ben-David *et al.* [9] is that "distributional **NP** search problems" with **P**-computable distributions are actually reducible to distributional decision problems in Dist($\mathbf{NP}, \mathbf{P}$-comp). Nonetheless, since we have not defined "distributional **NP** search problems" and the main subject of this thesis is decision problems, we state the result below without proof. The interested reader may refer to [9] for the definition of search problems and the proof of the theorem.

Theorem 5.5.10 [9] *Let* (R, μ) *be be a distributional* **NP** *search problem with* **P***-computable distribution*

μ. Then there is a distributional decision problem (D,ν) in $\mathrm{Aver}(\mathbf{NP},\mathbf{P}\text{-comp})$ *such that (R,μ) is bpp-tt-reducible to (B,ν).*

In worst-case complexity theory, it is easy to see that if $A \leq_T^{\mathrm{p}} C$ and $B \leq_T^{\mathrm{p}} C$, then $A \oplus B \leq_T^{\mathrm{p}} C$. The same is true in average-case complexity. In the next lemma, we prove this claim.

Recall from Section 3.5 the definition of $\mu \oplus \nu$ for two distributions μ and ν.

Lemma 5.5.11 *Let $\leq_\alpha \in \{\leq_m^{\mathrm{p}}, \leq_m^{\mathrm{avp}}, \leq_T^{\mathrm{p}}, \leq_T^{\mathrm{avp}}, \leq_T^{\mathrm{bpp}}, \leq_T^{\mathrm{avbpp}}\}$. If $(A,\mu_A) \leq_\alpha (C,\xi)$ and $(B,\mu_B) \leq_\alpha (C,\xi)$, then $(A \oplus B, \mu_A \oplus \mu_B) \leq_\alpha (C,\xi)$.*

Proof. We demonstrate only the case $\leq_T^{\mathrm{avbpp}}$. The other cases are somewhat similar. Let us assume that $(A,\mu_A) \leq_T^{\mathrm{avbpp}} (C,\xi)$ via M_A and $(B,\mu_B) \leq_T^{\mathrm{avbpp}} (C,\xi)$ via M_B. Define a new algorithm N: on input x, if $x = 0u$ for some u, then simulate M_A on input u; if $x = 1u$ for some u, then simulate M_B on u; otherwise, reject the input. For the sake of convenience, write ν for $\mu_A \oplus \mu_B$. Take a computational path y of the computation tree given by N on input x. Since N basically follows either M_A's computation or M_B's, we can determine the unique computation path, which is made by either M_A or M_B, corresponding to the computation path specified by random input s. We denote this computation path by $\tilde{y}$.

We show that $(A \oplus B,\nu) \leq_T^{\mathrm{avbpp}} (C,\xi)$ via N. By our assumption, there exist semi-distributions η_A and η_B such that $\mu_A \preceq^{\mathrm{avp}} \eta_A$, $\mu_B \preceq^{\mathrm{avp}} \eta_B$, $\hat{\xi}(z) \geq \hat{\eta}'_A(\{(u,s) \mid z \in Q(M_A,C,u,s)\})$, and $\hat{\xi}(z) \geq \hat{\eta}'_B(\{(u,s) \mid z \in Q(M_B,C,u,s)\})$ for all z, where η'_A and η'_B are the induced distributions from μ_A and $\Gamma_{M_A^C}$, and from μ_B and $\Gamma_{M_B^C}$, respectively. Let η be defined as

$$\hat{\eta}(x) = \begin{cases} \frac{1}{2} \cdot \hat{\eta}_A(u) & \text{if } x = 0u \text{ for some } u, \\ \frac{1}{2} \cdot \hat{\eta}_B(u) & \text{if } x = 1u \text{ for some } u, \\ 0 & \text{otherwise.} \end{cases}$$

It is obvious that η is a distribution. We also let $\eta = \eta_A \oplus \eta_B$. Let

$$p(x) = \begin{cases} p_A(u) & \text{if } x = 0u, \\ p_B(u) & \text{if } x = 1u, \\ 1 & \text{otherwise.} \end{cases}$$

We next show that $p(x) \cdot \hat{\eta}(x) \geq \hat{\nu}(x)$. This is seen as follows. Assume that x is of the form $0u$. The other case is similar:

$$\begin{aligned} p(x) \cdot \hat{\eta}(x) &= p_A(u) \cdot \frac{1}{2}\hat{\eta}_A(u) = \frac{1}{2} p_A(x) \cdot \hat{\eta}_A(u) \\ &\geq \frac{1}{2}\hat{\mu}_A(u) = \hat{\nu}(x). \end{aligned}$$

Finally, we show that $\hat{\xi}(z) \geq \hat{\eta}_\Gamma(\{(x,y) \mid z \in Q(N,C,x,y)\})$ for all z. For each string z, we have

$$\begin{aligned} \hat{\xi}(z) &\geq \frac{1}{2}\hat{\eta}'_A(\{(u,s) \mid z \in Q(M_A,C,u,s)\}) + \frac{1}{2}\hat{\eta}'_B(\{(u,s) \mid z \in Q(M_B,C,u,s)\}) \\ &= \hat{\eta}_\Gamma(\{(0u,s) \mid z \in Q(N,C,0u,s)\}) + \hat{\eta}_\Gamma(\{(1u,s) \mid z \in Q(N,C,1u,s)\}) \\ &= \hat{\eta}_\Gamma(\{x,s) \mid z \in Q(N,C,x,s)\}). \end{aligned}$$

This completes the proof. □

5.6 Structure of Reducibility

In this section, we shall take a close look at the structure of the randomized complexity class Dist(**NP**, **P**-comp) in the light of reducibility. First we shall review the results of Ben-David, Chor, Goldreich, and Luby [9]. They proved that if there is a hard distributional problem in Dist(**NP**, **P**-comp), there is also a problem which is in Dist(**NP**, **P**-comp) but not p-m-complete for Dist(**NP**, **P**-comp).

Proposition 5.6.1 **[9]** *Let $\mu \in$ strict-**REC**-comp and $D \in$ **REC**. Assume that there exists a time-constructible function p such that $p(n) \in n^{O(1)}$ and $|\mu(1^{p(i)}) - \mu(1^{p(j)})| \leq 2^{-i} + 2^{-j}$ for almost all $i, j \in \mathbb{N}$. If $(D, \mu) \notin \mathrm{Aver}(\mathbf{P}, *)$, then there exists a semi-distribution ν such that (i) $(D, \nu) \notin \mathrm{Aver}(\mathbf{P}, *)$, (ii) $(D, \mu) \not\leq^{\mathrm{p}}_T (D, \nu)$, (iii) $(D, \nu) \leq^{\mathrm{p}}_m (D, \mu)$, and (iv) ν is computed by an oracle log-space deterministic Turing machine relative to the function oracle μ.*

Proof. The proof proceeds by a so-called *slow diagonalization* technique. We first enumerate all quadruples $\langle M_i, N_i, c_i, d_i \rangle$ of a deterministic Turing machine M_i, a deterministic oracle Turing machine N_i, and natural numbers c_i and d_i. Write also D_i for $L(M_i)$ for brevity. Note that by our assumption, there exists a deterministic Turing machine which exactly computes $\hat{\mu}$.

For a finite sequence σ and an integer $k \geq 1$, let $\sigma(k)$ denote its kth element of σ. For a finite sequence σ of 0's and 1's, let

$$\nu(x) = \sigma(|x|) \cdot (\mu(x) - \mu(1^{|x|-1})) + \sum_{i<|x|} \sigma(i) \cdot (\mu(1^i) - \mu(0^i)).$$

It is easy to see that, for each x, $\hat{\nu}(x) = 0$ if $\sigma(|x|) = 0$, and $\hat{\nu}(x) = \hat{\mu}(x)$ otherwise. Note that the condition on μ that $|\mu(1^{p(i)}) - \mu(1^{p(j)})| \leq 2^{-i} + 2^{-j}$ will be used later to guarantee the existence of a "normalized" distribution of ν.

Let us consider the following subprogram $Cond(i, \sigma, t)$:

subroutine $Cond(i, \sigma, t)$
  if t steps are consumed during the following computation **then** return "no"
  if $i =$ "even" **then** let $i_o := \lfloor i/2 \rfloor$ and **go to** $(*)$
  let $i_0 = \lfloor (i-1)/2 \rfloor$ (since $i =$ "odd")
  check the following two conditions:
    (i) $D_{i_0} \cap \Sigma^{\leq|\sigma|} \neq D \cap \Sigma^{\leq|\sigma|}$; **and**
    (ii) $\sum_{x:0<|x|\leq|\sigma|} \mathrm{Time}_{M_{i_0}}(x)^{1/d_{i_0}} \cdot |x|^{-1} \cdot \hat{\nu}(x) > c_{i_0}$
  if either (i) or (ii) is true **then** return "yes" **else** return "no"
$(*)$ check the following three conditions:
    (i) $L(N_{i_0}, D_{i_0}) \cap \Sigma^{\leq|\sigma|} \neq D \cap \Sigma^{\leq|\sigma|}$;

(ii) $\sum_{x:0<|x|\le|\sigma|} \mathrm{Time}_{N_{i_0}}^{D_{i_0}}(x)^{1/d_{i_0}} \cdot |x|^{-1} \cdot \hat{\mu}(x) > c_{i_0}$; and

(iii) there exist two x, y of length $< |\sigma|$ such that $\hat{\mu}(x) > \hat{\nu}(y) = 0$, and $y \in Q(N_{i_0}, D_{i_0}, x)$

if either (i), (ii), or (iii) is true **then** return "yes" **else** return "no"

return.

Notice that if $Cond(i, \sigma, t)$ answers "yes," then so does $Cond(i, \tau, t)$ for all extensions τ of σ (i.e., $\tau \sqsupseteq \sigma$).

The following is the main body of the algorithm which computes σ:

begin deterministic algorithm for σ

input k

let $t := \lfloor \log k \rfloor$

for $i = 1$ **to** t **do**

exit the **for**-loop **when** t individual steps are executed in this **for**-loop

compute $\mu(1^i)$

if $\mu(1^i) > \frac{1}{2}$ **then go to** $(*)$

end-for

output $\sigma(k) := 1$ and **halt**

$(*)$ **for** $j = 1$ **to** t **do**

exit the **for**-loop **when** t individual steps are executed in this **for**-loop

re-compute $\sigma(j)$ (by a recursive call)

end-for

let J be the largest index j for which $\sigma(j)$ has been recomputed in the above **for**-loop

if no such J exists **then output** $\sigma(k) := 1$ and **halt**

set $\overline{\sigma}_J := \sigma(1)\sigma(2)\cdots\sigma(J)$

for $i = 1$ **to** J **do**

exit the **for**-loop **when** t individual steps are executed in this **for**-loop

call $Cond(i, \overline{\sigma}_J, t)$

if $Cond(i, \overline{\sigma}_J, t) =$ "no" **then exit** the **for**-loop

end-for

let I be the smallest index i for which $Cond(i, \overline{\sigma}_J, t)$ answers "no"

output $\sigma(k) := I \bmod 2$ and **halt**

end.

The **exit**-statement enforce a time bound t $(= \lfloor \log k \rfloor)$ on each each **for**-loop; thus, this algorithm requires at most $O(\log k)$ steps. Hence, σ is computed by using $O(\log k)$-space. By definition, ν is correctly computable using log-space with the help of μ as an oracle.

We shall show that this algorithm correctly computes $\sigma(k)$ on input k. We assume that there is no extension τ of σ such that $Cond(i, \tau, t)$ answers "yes." Assume that i is odd. Then τ is of the form $\sigma 1^m$ for some $m \in \mathbb{N}$ because $\sigma(j)$, $j > k$, is always set to be 1. We then have $D_{i_0} \cap \Sigma^{\le|\tau|} = D = \cap\Sigma^{\le|\tau|}$. This implies $L(M_{i_0}) = D$. Also we have $\sum_{x:0<|x|\le|\tau|} \mathrm{Time}_{M_{i_0}}(x)^{1/d_{i_0}} \cdot |x|^{-1} \cdot \hat{\nu}(x) \le c_{i_0}$. We remark that the

value $\hat{\nu}(x)$ equals $\hat{\mu}(x)$ if $|x| > |\sigma|$. Hence, for some constant $c > 0$, we have

$$\sum_{x:|x|>0} \mathrm{Time}_{M_{i_0}}(x)^{1/d_{i_0}} \cdot |x|^{-1} \cdot \hat{\mu}(x) \leq c + \sum_{x:|x|>0} \mathrm{Time}_{M_{i_0}}(x)^{1/d_{i_0}} \cdot |x|^{-1} \cdot \hat{\nu}(x) < c + c_{i_0} < \infty.$$

This implies that (D, μ) belongs to $\mathrm{Aver}(\mathbf{P}, *)$, a contradiction.

Now let us consider the other case that i is even. Notice that any extension τ of σ is of the form $\sigma 0^m$ for some $m \in \mathbb{N}$. Then $L(N_{i_0}, D_{i_0}) \cap \Sigma^{|\tau|} = D \cap \Sigma^{\leq |\tau|}$ for all extensions τ of σ. Thus, $L(N_{i_0}, D_{i_0}) = D$. Moreover, there are no pairs (x, y) such that $\hat{\mu}(x) > 0$, $\hat{\nu}(y) = 0$, and $y \in Q(N_{i_0}, D_{i_0}, x)$. Recall the definition of ν. Since $\tau = \sigma 0^m$, $m \in \mathbb{N}$, it follows that, for any y of length greater than $|\sigma|$, $\hat{\nu}(y) = 0$. This means that there is no x ($|x| > |\sigma|$) on which N_{i_0} queries strings of length greater than $|\sigma|$. In other words, $\| \bigcup_x Q(N_{i_0}, D_{i_0}, x) \|$ is finite. Let us define another Turing machine N' which simulates N_{i_0} on the same input with all oracle answers from D_{i_0} being encoded into its program. Clearly $L(N_{i_0}, D_{i_0}) = L(N)$, and $\mathrm{Time}_N(x) \leq c(\mathrm{Time}^{D_{i_0}}_{N_{i_0}}(x) + 1)$ for some absolute constant $c > 0$. Notice that $\sum_{x:0<|x|\leq|\tau|} |x|^{-1} \cdot \mathrm{Time}^{D_{i_0}}_{N_{i_0}}(x)^{1/d_{i_0}} \cdot \hat{\mu}(x) \leq c_{i_0}$ for all extensions τ. Hence, we have

$$\sum_{x:|x|>0} \frac{\mathrm{Time}_N(x)^{1/d_{i_0}}}{|x|} \cdot \hat{\mu}(x) \leq c' + c \cdot \sum_{x:|x|>0} \frac{\mathrm{Time}^{D_{i_0}}_{N_{i_0}}(x)^{1/d_{i_0}}}{|x|} \cdot \hat{\mu}(x) \leq c' + c \cdot c_{i_0}.$$

This again concludes that (D, μ) is in $\mathrm{Aver}(\mathbf{P}, *)$, a contradiction. Therefore, the algorithm for σ works properly.

By Corollary 4.3.3, we can normalize ν to a "full" distribution ν^* such that $\hat{\nu}^*(x) = \hat{\nu}(x)$ for all nonempty strings x. □

In a similar fashion, we can demonstrate the existence of a distributional problem that is not p-T-harder than a given problem, but can be p-m-reduced to the given problem.

Proposition 5.6.2 **[9]** *Let* $\mu \in$ strict-**REC**-comp *and* $D \in \mathbf{REC}$. *If* $(D, \mu) \notin \mathrm{Aver}(\mathbf{P}, *)$, *then there exists a set* E *such that (i)* $(E, \mu) \notin \mathrm{Aver}(\mathbf{P}, *)$, *(ii)* $(D, \mu) \not\leq^{\mathrm{p}}_T (E, \mu)$, *(iii)* $(E, \mu) \leq^{\mathrm{p}}_m (D, \mu)$, *and (iv)* $E \in \mathbf{L}^D$.

Proof. Similar to Theorem 5.6.1. □

Theorem 5.6.3 **[9]** *Assume that* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \not\subseteq \mathrm{Aver}(\mathbf{P}, *)$. *Then, there exists a distributional problem* (D, μ) *in the difference* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) - \mathrm{Aver}(\mathbf{P}, *)$ *which is not* $\leq^{\mathrm{p}}_m$*-complete for* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$.

Proof. Assume that $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \not\subseteq \mathrm{Aver}(\mathbf{P}, *)$. Then, no p-m-complete problem for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ belongs to $\mathrm{Aver}(\mathbf{P}, *)$. Let (D, μ) be any distributional problem which is p-m-complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. Since $(D, \mu) \notin \mathrm{Aver}(\mathbf{P}, *)$, Proposition 5.6.2 guarantees the existence of a set E such

that $(E,\mu) \notin \mathrm{Aver}(\mathbf{P},*)$, $(D,\mu) \not\leq_T^{\mathrm{p}} (E,\mu)$, and $(E,\mu) \leq_m^{\mathrm{p}} (D,\mu)$. As $(E,\mu) \leq_m^{\mathrm{p}} (D,\mu)$, the problem (E,μ) is in $\mathrm{Dist}(\mathbf{NP},\mathbf{P}\text{-comp})$.

Clearly (E,μ) is not p-m-complete for $\mathrm{Dist}(\mathbf{NP},\mathbf{P}\text{-comp})$. (Otherwise, $(E,\mu) \leq_m^{\mathrm{p}} (D,\mu)$, and thus $(E,\mu) \leq_T^{\mathrm{p}} (D,\mu)$, a contradiction.) □

We say that two distributional problems (D,μ) and (E,ν) are *incomparable* with respect to $\leq_T^{\mathrm{p}}$ if $(D,\mu) \not\leq_T^{\mathrm{p}} (E,\nu)$ and $(E,\nu) \not\leq_T^{\mathrm{p}} (D,\mu)$.

The following theorem leads to the existence of incomparable pairs in $\mathrm{Dist}(\mathbf{NP},\mathbf{P}\text{-comp})$ with respect to p-T-reductions.

Proposition 5.6.4 **[97]** *For every recursive set D not in $\mathbf{P}$, there exist two $\mathbf{P}$-computable distributions μ and ν such that $(D,\mu) \not\leq_T^{\mathrm{p}} (D,\nu)$ and $(D,\nu) \not\leq_T^{\mathrm{p}} (D,\mu)$.*

Proof. We use the slow diagonalization technique again. Let $\{M_i\}_{i\in\mathbb{N}}$ be a standard enumeration of all deterministic polynomial-time oracle Turing machines. We recall that s_i denotes the ith string in the standard order on Σ^*. Let σ,τ be infinite sequences of 0's and 1's. Let $\hat{\eta}(x) = 2^{-2|x|-1}$ for all x. This distribution is $\mathbf{P}$-computable. We next define a semi-distribution μ as

$$\mu(x) = \sigma(|x|)\cdot(\eta(x) - \eta(1^{|x|-1}) + \sum_{i<|x|} \sigma(i)\cdot(\eta(1^i) - \eta(0^i))$$

for any nonempty string x. Similarly, ν is defined by replacing σ with τ. Note that $\sigma(|x|) = \tau(|x|)$ if and only if $\hat{\mu}(x) = \hat{\nu}(x)$.

The infinite sequences σ and τ are computed by the following recursive procedure:

begin deterministic algorithm for $(\sigma(n),\tau(n))$
  input n
  if $n = 0$ **then output** $(\sigma(n),\tau(n)) := (1,1)$
    let $t = \lfloor \log n \rfloor$ (t does not change during $n \in \{2^m, 2^m+1, \ldots, 2^{m+1}-1\}$)
  for $j = 0$ **to** t **do**
    exit the **for**-loop **when** t steps have been executed
    recompute a pair $(\sigma(j),\tau(j))$ (by a recursive call)
  end-for
  let J be the largest index j for which $(\sigma(j),\tau(j))$ has been recomputed
  if no such J exists **then output** $(\sigma(n),\tau(n)) := (1,1)$
  for $i = 1$ **to** J **do**
    call $Cond(i,J,t)$
    if $Cond(i,J,t) =$ "no" **then** exit
  end-for
  let I be the smallest index i for which $Cond(i,J,t) =$ "no"
  (for the sake of convenience below, write $r(n) = I$)

if no such I exists **then output** $(\sigma(n), \tau(n)) := (1, 1)$

output $(\sigma(n), \tau(n)) := ((I+1) \bmod 2, I \bmod 2)$

end.

The subroutine $Cond(i, J, t)$ is as follows:

subroutine $Cond(i, J, t)$

if $i =$ "odd" **then** let $i_0 = \lfloor (i-1)/2 \rfloor$ and **go to** $(*)$

let $i_0 = \lfloor i/2 \rfloor$ (because $i =$ "even")

check the following two conditions:

(i) $L(M_{i_0}, D) \cap \Sigma^{\leq |\sigma|} \neq D \cap \Sigma^{\leq |\sigma|}$ or

(ii) there are $x, y < |\sigma|$ s.t. $\hat{\mu}(x) > \hat{\nu}(y) = 0$ and $y \in Q(M_{i_0}, D, x)$

if either (i) or (ii) is true **then return** "yes" **else return** "no"

$(*)$ check the following two conditions:

(i) $L(M_{i_0}, D) \cap \Sigma^{\leq |\sigma|} \neq D \cap \Sigma^{\leq |\sigma|}$ or

(ii) there are $x, y < |\sigma|$ s.t. $\hat{\nu}(x) > \hat{\mu}(y) = 0$ and $y \in Q(M_{i_0}, D, x)$

if either (i) or (ii) is true **then return** "yes" **else return** "no"

return.

Claim 14 *range*$(r) = \mathbb{N}$, *where* *range*$(r) = \{r(z) | z \in \mathbb{N}\}$.

Proof of Claim. Assume range$(r) \neq \mathbb{N}$. Take the minimal integer I such that $r(n) \leq I$ for all $n > 0$. Since I does not change, the algorithm always takes the same index I.

First consider the case that I is even. Let $i_0 = I/2$ and let n be sufficiently large. Notice that, for every x of length at least n, $\hat{\mu}(x) > 0$ and $\hat{\nu}(x) = 0$. Moreover, for every x, we have $M_{i_0}^D(x) = D(x)$, and $M_{i_0}^D(x)$ does not query any string y, where $\hat{\mu}(x) > \hat{\nu}(y) = 0$. Hence, $M_{i_0}^D$ computes D on all inputs, and it queries only strings of length smaller than n. This implies that D is in $\mathbf{P}$, which contradicts our assumption. The same argument also holds for the case where I is odd. ■

Therefore, μ and ν are well-defined.

Finally, we must normalize these two semi-distributions to obtain the desired distributions. It is not hard to see that μ and ν are log-space computable since, in each stage n, we quit the simulations after $\log n$ steps are made. From Corollary 4.3.3, it suffices to show that, for $p(n) = n + 1$, $|\mu(1^{p(i)}) - \mu(1^{p(j)})| \leq 2^{-i} + 2^{-j}$ holds for almost all $i, j \in \mathbb{N}$. Assume that $i \geq j$. Then,

$$\begin{aligned} |\mu(1^{p(i)}) - \mu(1^{p(j)})| &= \sum_{k=p(j)+1}^{p(i)} \hat{\mu}(\Sigma^k) \leq \sum_{k=j+2}^{i+1} \hat{\eta}(\Sigma^k) \\ &= \eta(1^{p(i)}) - \eta(1^{p(j)}) = (1 - 2^{-i}) - (1 - 2^{-j}) \\ &\leq 2^{-i} + 2^{-j} \end{aligned}$$

because either $\hat{\mu}(\Sigma^k) = \hat{\eta}(\Sigma^k)$ or $\hat{\mu}(\Sigma^k) = 0$. By Corollary 4.3.3, μ can be normalized by μ' such that $\hat{\mu}'(x) = \hat{\mu}(x)$ for all nonempty strings x. Thus, μ' is **P**-computable. In a similar way, we can show the existence of the **P**-computable distribution ν' which normalizes ν. Distributions μ' and ν' satisfy the conditions of the theorem. Thus, we complete the proof. □

Corollary 5.6.5 **[97]** *If* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \not\subseteq \mathrm{Aver}(\mathbf{P}, *)$, *then there is an incomparable pair in* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$.

Proof. Assume that $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \not\subseteq \mathrm{Aver}(\mathbf{P}, *)$. Let D be a set in $\mathbf{NP} - \mathbf{P}$. This set D exists because $\mathbf{P} = \mathbf{NP}$ implies $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \subseteq \mathrm{Aver}(\mathbf{P}, *)$. By Proposition 5.6.4, there exist two distributions $\mu, \nu \in \mathbf{P}\text{-comp}$ such that (D, μ) and (D, ν) are incomparable. Notice that (D, μ) and (D, ν) are in $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. □

5.7 Recent Topics

In this section, we shall discuss some issues which have emerged recently.

Randomized Many-One Reductions. The search for a better definition of "reductions" is an exciting field in average-case complexity theory, because the different choices of reducibility can lead to different worlds.

For example, we have seen that there is no flat distribution for which a distributional problem is p-m-complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ if $\mathbf{EXP} \neq \mathbf{NEXP}$. However, Gurevich introduced a random version of many-one reductions under which some flat distributions make their distributional problems "complete." A similar phenomenon occurs in the case of random reductions. Impagliazzo and Levin [44] demonstrated that distributional **NP**-search problems can be randomly reduced to distributional decision problems.

Here is the definition of random reductions. Recall the definition of dilations from Section 3.6.

Definition 5.7.1 **[12]** For two distributional problems (D, μ) and (E, ν), (D, μ) is polynomial-time randomly (or randomizing) reducible to (E, ν) if there exists a function f which is computable by a randomized algorithm in polynomial time on μ-average, and a dilation Γ such that (i) for any $(x, s) \in \Gamma$, $x \in A$ if and only if $f(x, s) \in B$, and (ii) f satisfies the domination condition. (N.B. f takes inputs of the form (x, s).)

Using this type of reduction, the following interesting claim can be proved.

Proposition 5.7.2 **[36, 109]** *There exits a flat distribution* μ *such that* (BHP, μ) *is complete for* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ *under polynomial-time random reducibility.*

Alternative Definition of Polynomial on the Average. Let us take a quick look at Levin's definition of "polynomial on the average." Recall that a function g is polynomial on μ-average if and only if $\sum_{x:x\neq\lambda} \frac{g(x)^\delta}{|x|} \cdot \hat{\mu}(x) < \infty$. One of the observations is that it allows us to deal with distributions that put too much weight on the first few strings. J. Cai and A. Selman [19] considered such distributions inappropriate for a coherent, general theory of average-case complexity. As an alternative, they propose a different notion of "polynomial on the average" which counts on the convergence rate of the expectation.

Definition 5.7.3 [19] A function g is "t on the μ-average" (in the sense of Cai and Selman) if, for all $n \geq 1$,

$$\sum_{x:|x|\geq n} \frac{t^{-1}(g(x))}{|x|}\hat{\mu}(x) \leq \hat{\mu}(\Sigma^* - \Sigma^{<n}).$$

As long as μ satisfies $\lambda n.\hat{\mu}(\{z \mid |z| \geq n\}) \in \Omega(n^{-k})$ for some constant $k > 0$, Levin's definition of "polynomial on μ-average" coincides with that of Cai and Selman [19]. Moreover, this notion of Cai and Selman is equivalent to the notion of "polynomial on average with respect to $\{\mu_{\geq n}\}_{n\in\mathbb{N}}$" in Definition 3.3.7.

C. Rackoff also suggests (reported in [19]) the following definition.

Definition 5.7.4 [19] A function g is "t on the μ-average" (in the sense of Rackoff) if, for all $n \geq 1$,

$$\sum_{x:|x|=n} \frac{t^{-1}(g(x))}{|x|}\hat{\mu}(x) \leq \hat{\mu}(\Sigma^n).$$

Rackoff's definition also provides the same notion of "polynomial on μ-average" given by Levin if μ satisfies $\lambda n.\hat{\mu}(\{z \mid |z| = n\}) \in \Omega(n^{-k})$ for some constant $k > 0$.

However, Cai and Selman's definition seems inappropriate when we consider the polynomial-time many-one reductions introduced in Section 5.2. Let us denote by CS-Aver$(\mathbf{P}, \mathcal{F})$ the collection of all distributional decision problems (A, μ) such that $\mu \in \mathcal{F}$, and A is computed by a deterministic Turing machine M which satisfies the condition that, for all $n > 0$, $\sum_{x:|x|\geq n} \frac{\mathrm{Time}_M(x)^{1/k}}{|x|}\hat{\mu}(x) \leq \hat{\mu}(\{z \mid |z| \geq n\})$ for some constant $k > 0$.

Belanger and Wang [7] showed that CS-Aver$(\mathbf{P}, *)$ is not closed under $\leq_m^{\mathrm{p}}$-reductions or $\leq_m^{\mathrm{avp}}$-reductions with one-one reduction functions. Hence, CS-Aver$(\mathbf{P}, *)$ is properly included in Aver$(\mathbf{P}, *)$.

Theorem 5.7.5 **[7]** *CS-Aver$(\mathbf{P}, *)$ is not closed downward under $\leq_m^{\mathrm{p}}$-reductions or $\leq_m^{\mathrm{avp}}$-reductions with one-one reduction functions.*

Chapter 6

Average Case Hierarchies

6.1 Introduction

One of the most novel contributions to worst-case complexity theory is the introduction of the *polynomial-time hierarchy* (or *polynomial hierarchy*) by Meyer and Stockmeyer in 1972. This worst-case hierarchy is built from **P** and **NP** in a way similar to how Kleene constructed his arithmetical hierarchy above recursively enumerable sets. Notably, the classes of the hierarchy lies between **P** and **PSPACE**. The construction is such that the kth level of the polynomial-time hierarchy is defined to be the collection of all sets, each of which can be recognized by a polynomial-time deterministic/nondeterministic oracle Turing machine relative to some sets in the $(k-1)$th level of the hierarchy.

The polynomial-time hierarchy has been studied extensively for over two decades with many intriguing results. For example, if the Boolean hierarchy over **NP** collapses, then so does the polynomial-time hierarchy [47], and if the polynomial-time hierarchy collapses, then the low and high hierarchies in **NP** collapse [89]. Many **NP**-hard problems are classified into various levels in the polynomial-time hierarchy.

This chapter will build average-case analogues of the polynomial-time hierarchy among distributional decision problems. To introduce an average polynomial-time hierarchy, we shall begin with the relativization of the fundamental average-case complexity classes Aver$(\mathbf{P}, \mathcal{F})$ and Aver$(\mathbf{NP}, \mathcal{F})$, relative to a distributional problem (E, ν).

A relativization of the class Aver$(\mathbf{P}, \mathcal{F})$ to an oracle problem (E, ν) is naturally induced from the average polynomial-time deterministic Turing reducibility defined in Section 5.2. A similar approach toward a relativized Aver$(\mathbf{NP}, \mathcal{F})$ is taken by Schuler and Yamakami [97] based on the model of clocked nondeterministic Turing machines. Here we shall define a relativized Aver$(\mathbf{NP}, \mathcal{F})$ in a slightly different way. In Section 5.2, the relativization of Aver$(\mathbf{P}, \mathcal{F})$, Aver$(\mathbf{BPP}, \mathcal{F})$, Aver$(\mathbf{NP}, \mathcal{F})$, and Aver$(\mathbf{PSPACE}, \mathcal{F})$ will be formulated in terms of various restrictions on average polynomial-time oracle Turing machines together with weaker domination conditions for those machines.

In relativized worlds, we can see a desirable separation of Aver$(\mathbf{NP}, \mathbf{P}\text{-comp})$ and Aver$(\mathbf{P}, \mathbf{P}\text{-comp})$, and

a collapse between Aver($\mathbf{P}$, $\mathbf{P}$-comp) and Aver($\mathbf{PSPACE}$, $\mathbf{P}$-comp).

In Section 6.4, using Turing reducibility, we shall introduce an average-case version of the polynomial-time hierarchy, average polynomial-time hierarchy under a set $\mathcal{F}$ of distributions $\{\mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F}), \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F}), \mathrm{Aver}(\mathbf{\Pi}_k^{\mathrm{p}}, \mathcal{F}) \mid k > 0\}$.

Another effective way of characterizing the kth level of the polynomial-time hierarchy is to use polynomial-time alternating Turing machines with k-alternation. This worst-case characterization is very suggestive and enables us to build another type of average polynomial-time hierarchy using "average" polynomial-time alternating Turing machines with constant-alternation. We call this average-case hierarchy the *average polynomial-time alternating hierarchy*. Section 6.5 will formally introduce an average-case alternation hierarchy based on average polynomial-time alternating Turing machines with constant-alternation. As the reader may perceive, however, these two types of average-case hierarchies are unlikely to coincide.

We have seen in the previous chapter that if Dist($\mathbf{NP}$, $\mathbf{P}$-comp) is not included in Aver($\mathbf{P}$, $\mathbf{P}$-comp), then there are distributional problems which are in Dist($\mathbf{NP}$, $\mathbf{P}$-comp) but which are not p-m-complete. This indicates the possibility of a large gap between Aver($\mathbf{P}$, $\mathbf{P}$-comp) and Aver($\mathbf{NP}$, $\mathbf{P}$-comp). One approach of refining this gap involves constructing a hierarchical structure within Aver($\mathbf{NP}$, $\mathbf{P}$-comp) and sorting out all distributional problems in Aver($\mathbf{NP}$, $\mathbf{P}$-comp) into various levels of this hierarchy. This approach was taken by Schöning in worst-case complexity theory in 1980, and the hierarchies within $\mathbf{NP}$ are called the *high and low hierarchies*. A natural average-case version of the low hierarchy, called the *average low hierarchy* within Aver($\mathbf{NP}$, $\mathcal{F}$), will be introduced.

Major Contributions. In this chapter, we introduce several new hierarchies.

In Proposition 6.2.8, it follows by the self-reducibility of RBHP that Dist($\mathbf{NP}$, $\mathbf{P}$-comp) $\subseteq$ Aver($\mathbf{BPP}$, $*$) implies Dist($\mathbf{NP}$, $\mathbf{P}$-comp) $\subseteq$ Aver($\mathbf{RP}$, $*$).

The notion of relativization originally comes from Schuler and Yamakami [97]; however, the relativizations of the classes Aver($\mathbf{NP}$, $\mathcal{F}$) and Aver($\mathbf{BPP}$, $\mathcal{F}$) treated in this chapter are different from those in [97].

Proposition 6.3.8 shows that Aver($\mathbf{BPP}$, $\mathcal{F}$) relative to Aver($\mathbf{BPP}$, $*$) collapses to Aver($\mathbf{BPP}$, $*$) for any set $\mathcal{F}$ of distributions.

Proposition 6.3.11 shows basic properties of relativized Aver($\mathbf{NP}$, $\mathcal{F}$), such as reflexivity and transitivity. In particular, if (A, μ) is in Aver($\mathbf{P}$, $*$)$^{(B,\nu)}$, then all problems in Aver($\mathbf{NP}$, $\mathcal{F}$)$^{(A,\mu)}$ belong to Aver($\mathbf{NP}$, $\mathcal{F}$)$^{(B,\nu)}$.

Theorem 6.3.12 demonstrates an oracle separation between Aver($\mathbf{P}$, $\mathbf{P}$-comp) based on a tally oracle construction given by Baker, Gill, and Solovay [3], whereas Theorem 6.3.17 shows a collapse of Aver($\mathbf{PSPACE}$, $\mathbf{P}$-comp) to Aver($\mathbf{P}$, $\mathbf{P}$-comp) in a relativized world. The proof uses a relativized version of the randomized bounded halting problem.

Proposition 6.3.19 shows that Aver($\mathbf{PSPACE}$, $\mathcal{F}$) relative to Aver($\mathbf{PSPACE}$, $*$) collapses to Aver($\mathbf{PSPACE}$, $\mathcal{F}$) for any set $\mathcal{F}$ of distributions.

Basic inclusion relationships, such as Aver($\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F}$) $\subseteq$ Aver($\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F}$) $\subseteq$ Aver($\mathbf{\Delta}_{k+1}^{\mathrm{p}}, \mathcal{F}$), among classes in the average polynomial-time hierarchy are shown in Proposition 6.4.4.

Proposition 6.4.6 shows that a collapse of two levels of the average polynomial-time hierarchy causes any higher level of the hierarchy to collapse.

An important new idea is the use of of alternating machines to build another average-case version of the polynomial-time hierarchy. Theorem 6.5.2 gives an oracle characterization of the average polynomial-time alternating hierarchy; for example, $\mathrm{Aver}(\mathbf{A}\boldsymbol{\Delta}^{\mathrm{p}}_{k+1}, \mathcal{F})$ is equivalent to the class $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$ relative to $\mathrm{Dist}(\boldsymbol{\Sigma}^{\mathrm{p}}_{k}, *)$. A similar characterization holds for $\mathrm{Aver}(\mathbf{A}\boldsymbol{\Sigma}^{\mathrm{p}}_{k+1}, \mathcal{F})$.

In Proposition 6.5.4, it is shown that the class $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$ is included in the second level of the average polynomial-time alternation hierarchy.

Proposition 6.4.9 and Theorem 6.4.11 show that many basic average complexity classes, such as $\mathrm{Aver}(\mathbf{P}, *)$ and $\mathrm{Aver}(\mathbf{NP}, *)$, have the sparse interpolation property.

6.2 Distributional Polynomial-Time Hierarchy

In the previous chapter, we showed that several important distributional decision problems, such as the bounded halting problem RBHP, are p-m-complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. This section will extend these completeness results to the class $\mathrm{Dist}(\boldsymbol{\Sigma}^{\mathrm{p}}_{k}, \mathbf{P}\text{-comp})$, the kth level of the "distributional polynomial-time hierarchy under $\mathbf{P}$-comp."

6.2.1 Definition of Hierarchy

Randomized complexity classes are the simplest randomization of existing worst-case complexity classes. As with the polynomial-time hierarchy, we can consider its natural counterpart of the polynomial-time hierarchy, the *distributional polynomial-time hierarchy*, under a given set of distributions.

We begin with its formal definition.

Definition 6.2.1 Let $\mathcal{F}$ be a set of distributions. A *distributional polynomial-time hierarchy under* $\mathcal{F}$ consists of the following classes: $\mathrm{Dist}(\boldsymbol{\Delta}^{\mathrm{p}}_{k}, \mathcal{F})$, $\mathrm{Dist}(\boldsymbol{\Sigma}^{\mathrm{p}}_{k}, \mathcal{F})$, and $\mathrm{Dist}(\boldsymbol{\Pi}^{\mathrm{p}}_{k}, \mathcal{F})$ for all $k \geq 0$.

We next demonstrate the existence of p-m-complete problems for each class $\mathrm{Dist}(\boldsymbol{\Sigma}^{\mathrm{p}}_{k}, \mathbf{P}\text{-comp})$. We generalize the randomized bounded halting problem RBHP, seen earlier, to the kth level randomized bounded halting problem RBHP^k. The *kth level randomized bounded halting problem* RBHP^k is the distributional problem $(\mathrm{BHP}^k, \mu_{\mathrm{BHP}})$ defined in the following fashion. Let μ_{BHP} be the same distribution as in Section 5.3. Assuming that $\{M_i\}_{i \in \mathbb{N}}$ is an effective enumeration of all nondeterministic oracle Turing machines, we define

$$\mathrm{BHP}(A) = \{\langle i, x, 1^n \rangle \mid M_i^A \text{ accepts } x \text{ in less than } n \text{ steps } \},$$

and then set $\mathrm{BHP}^1 = \mathrm{BHP}(\emptyset)$ and $\mathrm{BHP}^{k+1} = \mathrm{BHP}(\mathrm{BHP}^k)$ for $k \geq 1$.

Next theorem proves that, for each k, the problem RBHP^k is p-m-complete for $\mathrm{Dist}(\boldsymbol{\Sigma}^{\mathrm{p}}_{k}, \mathbf{P}\text{-comp})$. Note that the case $k = 1$ has been already shown as in Theorem 5.3.2.

Theorem 6.2.2 **[97]** *For any $k > 0$, RBHP^k is $\leq^{\mathrm{p}}_m$-complete for $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathbf{P}\text{-comp})$.*

Proof. The general case $k > 1$ is very similar to the base case $k = 1$ of Theorem 5.3.2. We note that BHP^k is $\mathbf{\Sigma}^{\mathrm{p}}_k$-complete.

For every set $D \in \mathbf{\Sigma}^{\mathrm{p}}_k$ and every distribution $\mu \in \mathbf{P}$-comp, we shall show that $(D, \mu) \leq^{\mathrm{p}}_m (\mathrm{BHP}^k, \mu_{\mathrm{BHP}})$. Notice that BHP^{k-1} is p-m-complete for $\mathbf{\Sigma}^{\mathrm{p}}_{k-1}$, and as a consequence, there exists a polynomial-time nondeterministic oracle Turing machine M computing D with oracle BHP^{k-1}.

Let g be the function g defined in Lemma 4.2.7(2). The function g satisfies $\hat{\mu}(x) < 2^{-|g(x)|+2}$. Now let i be an index such that $L(M_i) = L(M)$. Let p be a polynomial time bound of M_i. For the reduction f from D to BHP^k, we set $f(x) = \langle s_i, g(x), 1^{p(|x|)} \rangle$. Clearly f is one-one and witnesses the reduction $D \leq^{\mathrm{p}}_m \mathrm{BHP}^k$ because $x \in D$ holds exactly when M_i with oracle BHP^{k-1} accepts x within $p(|x|)$ steps. To check the domination condition for f, we simply follow a argument similar to that used in the proof of Theorem 5.3.2. Thus, we obtain $(D, \mu) \leq^{\mathrm{p}}_m (\mathrm{BHP}^k, \mu_{\mathrm{BHP}})$. □

6.2.2 Self-Reducibility

Meyer and Paterson [71] have introduced the notion of *(polynomial-time Turing) self-reducibility* into worst-case complexity theory.

All known **NP**-complete problems are self-reducible, and every self-reducible set belongs to **PSPACE**. It is natural to ask whether the notion of self-reducibility has a counterpart in distributional decision problems. We shall indeed give in this section the formal definition of an average-case version of self-reducibility and show the existence of self-reducible sets in each level of the average polynomial-time hierarchy.

We begin by defining the important concept of a **P**-computable, OK partial order on the set Σ^*.

Definition 6.2.3 (OK Partial Order) Let $<$ be a partial order.

1. A partial order $<$ is *polynomial-time computable* (**P**-computable, for short) if there exists a polynomial-time deterministic Turing machine M such that, for every pair (x, y), $x < y$ if and only if $M(x, y) = 1$.

2. A partial order $<$ is *OK* if there exists a polynomial p such that

   (i) every strictly descending chain is finite and is polynomial in the length of its maximum element; i.e., if $x_1 > x_2 > \cdots > x_{k-1} > x_k$ is a strictly descending chain starting from x_1, then $k \leq p(|x_1|)$, and

   (ii) for every pair (x, y), $x < y$ implies $|x| \leq p(|y|)$.

For example, let us write $x < y$ to mean $|x| < |y|$ for any pair (x, y). Then the relation $<$ becomes a **P**-computable, OK *total* order on Σ^*.

The notion of self-reducibility is now stated as follows.

Definition 6.2.4 (Self-Reducibility) [97] A distributional decision problem (D, μ) is *(polynomial-time Turing) self-reducible* if there exist an OK partial order and a deterministic oracle Turing machine M such that $(D, \mu) \leq_T^{\mathrm{p}} (D, \mu)$ via M, and for every input x, all query strings in the computation of M on input x are smaller than x with respect to the partial order. The machine M is called a *self-reducing machine* for (D, μ).

Lemma 6.2.5 *1. Every distributional decision problem in* $\mathrm{Dist}(\mathbf{P}, *)$ *is self-reducible.*

2. Every self-reducible distributional decision problem is in $\mathrm{Dist}(\mathbf{PSPACE}, *)$.

Proof. (1) For any distributional problem (D, μ) in $\mathrm{Dist}(\mathbf{P}, *)$, there exists a deterministic Turing machine M which computes (D, μ) without any queries. This machine reduces (D, μ) to (D, μ). By the definition of self-reducibility, (D, μ) turns out to be self-reducible.

(2) Assume that (D, μ) is self-reducible. Let us take a self-reducing machine M for (D, μ). We shall define a Turing machine N which computes D using polynomial space. Below we describe an algorithm for N using a recursive call.

```
begin deterministic algorithm for N
    input x
    simulate M on input x
    while simulation do
(*)     if M makes a query z then simulate N on input z
            if M reaches a halting configuration then output M(x) and halt
    end-while
end.
```

We must prove that N uses only polynomial space. Let us analyze the query process of N on input x. Notice that the space used by the machine M is p-bounded, and thus there are at most exponentially-many different configurations. Remember that this number is independent of the choice of oracle. Let us assume that in the first round of the simulation of M, M queries at most exponentially-many strings, which are in $Q(M, D, x)$. The recursive protocol in the line $(*)$ brings another round of the simulation of M on each input taken from the set $Q(M, D, x)$. Fix an arbitrary string $z^{(1)}$ in $Q(M, D, x)$. In the second round, M makes another set of queries, $Q(M, D, z^{(1)})$. To go into the third round, we fix a query string $z^{(2)}$ in $Q(M, D, z^{(1)})$ and then consider the set $Q(M, D, z^{(2)})$ of query strings. Recursively, we keep fixing a query string $z^{(i)}$ and then start another round of the simulation of M on the input $z^{(i)}$. This process proceeds until M does not query any strings.

Now let us consider an arbitrary sequence $z^{(1)}, z^{(2)}, \ldots, z^{(i)}, \cdots$ taken by the above procedure. Since M is a self-reducing machine, this sequence makes a strictly descending chain with respect to the given OK partial order. This implies that the length of the chain is bounded above by a polynomial in $|x|$, and the length of each string $z^{(i)}$ is also bounded by a polynomial in $|x|$. Let p be such a polynomial. As a result, N

can simulate M using $p(|x|)$ blocks on a working tape, each of which is used to store one of the strings $z^{(i)}$ being queried in each ith round. Therefore, N needs only polynomial space. □

Moreover, the set of all self-reducible problems is closed under p-isomorphism; i.e., if (D, μ) is p-isomorphic to some distributional problem which is self-reducible, then (D, μ) is self-reducible.

Lemma 6.2.6 *Let (D, μ) and (E, ν) be any distributional problems. If (D, μ) is p-isomorphic to (E, ν) and (E, ν) is self-reducible, then (D, μ) is self-reducible.*

Proof. Let f be a p-isomorphism from (D, μ) to (E, ν), and let M be a self-reducing machine for (E, ν). By the definition of p-isomorphism, f is **P**-computable and p-invertible; that is, f^{-1} is also **P**-computable.

Since $(D, \mu) \leq_1^{\mathrm{p}} (E, \nu)$ via f, there exists a polynomial p such that $p(|x|) \cdot \hat{\nu}(f(x)) \geq \hat{\mu}(x)$ for all x. Similarly, f^{-1} reduces (E, ν) to (D, μ), and thus there is a positive, strictly increasing polynomial q such that $q(|f(x)|) \cdot \hat{\mu}(x) \geq \hat{\nu}(f(x))$ for all x.

We wish to construct another self-reducing machine N for (D, μ). Let us define the machine N as follows:

begin deterministic algorithm for N with oracle
  input x
  start the simulation of M on input $f(x)$
  while simulation **do**
    if M queries z **then** query $f^{-1}(z)$ to oracle
    if M reaches an accepting configuration **then accept**
  end-while
  accept
end.

Recall that $D(x) = E(f(x))$ and $D(f^{-1}(z)) = E(z)$ for all strings x and z. From these equations, N can reduce D to D in polynomial time. Furthermore, there exists a strictly increasing polynomial q' such that if z is a query string made by N on input x, then $|f(z)| \leq q'(|x|)$, since f and f^{-1} are both p-honest and p-bounded.

Next we shall check the domination condition for N. By the domination condition for M, we can find a semi-distribution ξ and a polynomial p' such that $p'(|x|) \cdot \hat{\xi}(x) \geq \hat{\nu}(x)$ and $\hat{\nu}(z) \geq \hat{\xi}(\{x \mid z \in Q(M, E, x)\})$ for all x and z. We define the semi-distribution η by $\hat{\eta}(x) = \hat{\xi}(f(x))/s(|x|)$, where $s = q \circ q'$. We first show that $\mu \preceq^{\mathrm{p}} \eta$. This is seen as follows. For each x, we have:

$$\hat{\mu}(x) \leq p(|x|) \cdot \hat{\nu}(f(x)) \leq p(|x|)p'(|f(x)|) \cdot \hat{\xi}(f(x)) \leq p(|x|)p'(|f(x)|)s(|x|) \cdot \hat{\eta}(x).$$

From these inequalities, we set $t(z) = p(z)p'(f(z))s(x)$, and thus $t(x) \cdot \hat{\eta}(x) \geq \hat{\mu}(x)$.

To see that $\hat{\mu}(z) \geq \hat{\eta}(\{x \mid z \in Q(N,D,x)\})$, we notice that

$$\hat{\eta}(\{x \mid z \in Q(N,D,x)\}) \leq \sum_{x:z \in Q(N,D,x)} \frac{\hat{\xi}(f(x))}{s(|x|)} \leq \sum_{x:z \in Q(N,D,x)} \frac{\hat{\xi}(f(x))}{q(f(z))} \leq \frac{1}{q(f(z))} \cdot \sum_{x:z \in Q(N,D,x)} \hat{\xi}(f(x)).$$

The term $\sum_{x:z \in Q(N,D,x)} \hat{\xi}(f(x))$ is equivalent to the term $\hat{\xi}(\{y \mid f(z) \in Q(M,E,y)\}$, which is bounded by $\hat{\nu}(f(z))$. Therefore, it follows that

$$\hat{\eta}(\{x \mid z \in Q(N,D,x)\}) \leq \frac{\hat{\nu}(f(z))}{q(f(z))} \leq \hat{\mu}(z).$$

This completes the proof. □

One of the classical self-reducible **NP**-complete problems is the *satisfiability problem*, SAT. However, we do not know a simple distribution μ such that (SAT, μ) is $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$-complete. Moreover, as shown by Franco and Paull [25], SAT is deterministically computable in polynomial-time on the average under some natural distribution. Instead, we consider a skew version of the kth level randomized bounded halting problem, skewRBHP^k, which is p-isomorphic to RBHP^k. We want to show that skewRBHP^k is self-reducible. By Lemma 6.2.6, this implies the self-reducibility of RBHP^k.

First we define the kth level randomized skew bounded halting problem skewRBHP^k as the distributional decision problem $(\mathrm{BHP}_{skew}, \mu_{\mathrm{BHP}_{skew}})$. Assuming that all nondeterministic Turing machines take at most two nondeterministic choices at every step, BHP_{skew} is the collection of strings of the form $\langle s_i, x, y, 1^t \rangle$, $|y| \leq t$, such that, on input x, M_i with oracle BHP^{k-1} deterministically follows the computation path specified by y during the first $|y|$ nondeterministic choices and then halts in an accepting state within t steps. The distribution $\mu_{\mathrm{BHP}_{skew}}$ is defined by:

$$\hat{\mu}_{\mathrm{BHP}_{skew}}(s_i, x, y, t) = \hat{\mu}_{\mathrm{BHP}}(s_i, x, t) \cdot \hat{\nu}_{\mathrm{stand}}(y).$$

Theorem 6.2.7 **[97]** *For each $k > 0$, RBHP^k is self-reducible.*

Proof. As mentioned above, our strategy is to show that (i) skewRBHP^k is p-isomorphic to RBHP^k, and (ii) skewRBHP^k is self-reducible.

We first sketch the proof of statement (i). The proof that skewRBHP$^k \leq_1^{\mathrm{p}}$ RBHP^k via a **P**-computable, p-invertible reduction function is similar to the proof of Theorem 5.3.2. The other direction RBHP$^k \leq_1^{\mathrm{p}}$ skewRBHP^k is shown by considering the reduction function f defined by

$$f(s_i, x, t) = \langle s_i, x, \lambda, t \rangle$$

for all triplets (s_i, x, t). The function f is one-one, **P**-computable, and p-invertible.

Next we prove statement (ii); that is, skewRBHP^k is self-reducible. Let us define the binary relation $<$ as follows:

$$(s_i, x, y, t) < (s_j, x', y', t') \Longleftrightarrow i = j \wedge x = x' \wedge y' \sqsubseteq y \wedge t = t' \wedge |y| \leq |t'|.$$

It is easy to see that the relation $<$ is **P**-computable and that it is an OK partial order because of the upper bound $|t'|$ on the size of y.

Next let us consider the following oracle Turing machine N:

begin deterministic algorithm for N
  input $\langle i, x, y, t\rangle$
  if t is not of the form 1^m **then reject**
  if $|y| > |t|$ **then reject**
  decode the code s_i and recover the machine M_i
  simulate M_i on input x deterministically following the computation path
    encoded by y until the machine either exhausts $|t|$ steps or
    makes the $(|y|+1)$th nondeterministic choice
  if either M_i reaches a halting state or it does not halt within $|t|$ steps **then** go to $(*)$
  query $\langle i, x, y0, t\rangle$ and $\langle i, x, y1, t\rangle$ to oracle
  if one of the strings belongs to oracle **then accept else reject**
$(*)$ **if** M_i accepts x within $|t|$ steps **then accept else reject**
end.

We wish to show that N is self-reducing machine. Clearly N reduces BHP^k_{skew} to BHP^k_{skew} by querying only strings which are smaller than input with respect to $<$. To complete the proof, we must check the domination condition for N. Notice that two query strings $\langle s_i, x, y0, t\rangle$ and $\langle s_i, x, y1, t\rangle$ uniquely correspond to input $\langle s_i, x, y, t\rangle$. Let us consider the query string $\langle s_i, x, y0, t\rangle$. The probability $\hat{\mu}_{\mathrm{BHP}_{skew}}(s_i, x, y0, t)$ is bounded by:

$$\begin{aligned}
8 \cdot \hat{\mu}_{\mathrm{BHP}_{skew}}(s_i, x, y0, t) &= 8 \cdot \hat{\mu}_{\mathrm{BHP}}(s_i, x, t) \cdot \hat{\nu}_{\mathrm{stand}}(y0) \\
&= 8 \cdot \hat{\mu}_{\mathrm{BHP}}(s_i, x, t) \cdot 2^{-2\mathrm{llog}(|y0|)-|y0|-1} \\
&\geq 8 \cdot \hat{\mu}_{\mathrm{BHP}}(s_i, x, t) \cdot 2^{-3} \cdot 2^{-2\mathrm{llog}(|y|)-|y|-1} \\
&= \hat{\mu}_{\mathrm{BHP}_{skew}}(s_i, x, y, t) \\
&= \hat{\mu}_{\mathrm{BHP}_{skew}}(\{w \mid (s_i, x, y0, t) \in Q(N, \mathrm{BHP}^k_{skew}, w)\})
\end{aligned}$$

since $\mathrm{llog}(n+1) \leq \mathrm{llog}(n) + 1$ for all $n \in \mathbb{N}$. A similar inequality holds for $\hat{\mu}_{\mathrm{BHP}_{skew}}(s_i, x, y1, t)$. Therefore, N is a self-reducing machine. □

As stated before, Wang and Belanger [112] show that most known distributional problems complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ are p-isomorphic to each other. From Theorem 6.2.7, we immediately conclude that most known p-m-complete problems for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ are self-reducible. Nevertheless, we do not know whether *all* p-m-complete problems for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ are self-reducible. If the Isomorphism Conjecture (in Section 5.3) is true, then all distributional problems in $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ obviously become self-reducible.

Finally in this section, we shall demonstrate an application of self-reducibility. In worst-case complexity theory, Ko [52] and Zachos [122] use self-reducible sets to show that $\mathbf{NP} \subseteq \mathbf{BPP}$ implies $\mathbf{NP} = \mathbf{RP}$. A similar argument can be carried out in the average-case setting.

Proposition 6.2.8 $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \subseteq \mathrm{Aver}(\mathbf{BPP}, *)$ *if and only if* $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \subseteq \mathrm{Aver}(\mathbf{RP}, *)$.

Proof. Let us assume that $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ is included in $\mathrm{Aver}(\mathbf{BPP}, *)$. We have seen that skewRBHP is self-reducible and complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$. It is thus enough to show that skewRBHP belongs to $\mathrm{Aver}(\mathbf{RP}, *)$ because the class $\mathrm{Aver}(\mathbf{RP}, *)$ is closed under p-m-reductions.

Let M be a self-reducing machine for skewRBHP which runs in polynomial time, and a $\mathbf{P}$-computable, partial OK order $\leq$ for self-reducibility. Let p be a polynomial such that $\mathrm{Time}_M(x) \leq p(|x|)$ for all x. Let us consider the self-reducing tree of skewRBHP. By our assumption, there is a bounded-error probabilistic Turing machine M_0 which recognizes BHP in polynomial time on μ_{BHP}-average. Benefited by the Amplification Lemma, we can assume that $\mathbf{Pr}_{M_0}[M_0(z) = \mathrm{BHP}(z)] \geq 1 - 2^{-p(z)-3}$ for all z.

On input $\langle s_i, x, y, 1^m \rangle$, each node of the self-reducing tree starting from the root $\langle s_i, x, 1^m \rangle$ describes a nondeterministic choice made by the machine M_i on input x. To determine the correct outcome of the machine, we probabilistically trace the self-reducing tree along each path as a series of nondeterministic choices; when we reach a leaf, we check whether the machine reaches an accepting configuration. This last step is done without any error, and thus we have a one-sided error randomized algorithm.

Therefore, RBHP is in $\mathrm{Aver}(\mathbf{RP}, *)$. □

6.3 Relativization of Average Complexity Classes

In 1975, Baker, Gill, and Solovay [3] initiated a study of relativized complexity classes. Early studies revealed possible worlds in which $\mathbf{P} = \mathbf{NP} = \text{co-}\mathbf{NP}$, $\mathbf{P} \neq \mathbf{NP} = \text{co-}\mathbf{NP}$, or $\mathbf{P} \neq \mathbf{NP} \neq \text{co-}\mathbf{NP}$ holds. This appalling phenomenon clearly reflects the difference in computational power between determinism and nondeterminism.

In Chapter 5, we have already seen "relativized computation" in terms of average-case versions of Turing reducibility. Naturally we can expand our boundary to relativized worlds. This section will introduce four relativized average-case complexity classes: $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$, $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$, $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$, and $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$.

6.3.1 Relativized $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$

The concept of relativization is another way of viewing Turing reducibility. We have already seen two types of Turing reducibilities: deterministic Turing reducibility and bounded-error probabilistic Turing reducibility. Here we shall introduce the notation $\mathrm{Aver}(\mathbf{P}, \mathcal{F})^{(B,\nu)}$ to denote the collection of distributional problems which are avp-T-reducible to a given distributional problem (B, ν), analogous to the notation $\mathbf{P}^B$ in worst-case complexity theory.

Now let us introduce a relativization of the fundamental average classes $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$ and $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$.

Definition 6.3.1 (Relativized $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$) [97] Let (B, ν) be a distributional decision problem. For

a set $\mathcal{F}$ of distributions, denote by $\mathrm{Aver}(\mathbf{P},\mathcal{F})^{(B,\nu)}$ the collection of all distributional decision problems (A,μ) such that there exist a deterministic oracle Turing machine M and a semi-distribution η satisfying the following conditions:

(i) (Efficiency) M with oracle B is polynomial-time bounded on μ-average;

(ii) (Validity) $A = L(M,B)$; and

(iii) (Domination) $\mu \preceq^{\mathrm{avp}} \eta$ and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{x \mid z \in Q(M,B,x)\})$.

To improve readability, we simply say that the oracle machine M *computes* A *with oracle* (B,ν) *in polynomial time on* μ*-average* if Conditions (i)-(iii) are witnessed by M with some semi-distribution η.

We remark that it is possible to introduce a weaker type of relativization using p-T-reductions (see, e.g., [97]). However, we believe that average polynomial-time Turing reductions are a better choice when discussing average-case complexity, because the class $\mathrm{Aver}(\mathbf{P},*)$ is closed under avp-T-reductions.

Proposition 6.3.2 **[97]** *Let (A,μ) and (B,ν) be any distributional decision problems, and let $\mathcal{F}$ be any set of distributions.*

1. $(B,\nu) \in \mathrm{Aver}(\mathbf{P},\mathcal{F})^{(B,\nu)}$.
2. *If* $(A,\mu) \in \mathrm{Aver}(\mathbf{P},*)^{(B,\nu)}$, *then* $\mathrm{Aver}(\mathbf{P},\mathcal{F})^{(A,\mu)} \subseteq \mathrm{Aver}(\mathbf{P},\mathcal{F})^{(B,\nu)}$.

Proof. (1) Consider the oracle Turing machine that queries the input string to oracle and then accepts it exactly when it is in the oracle. (2) This is another interpretation of the transitivity property of avp-T-reducibility. □

Finally we shall extend Definition 6.3.1 from a single oracle problem to a class of oracle problems.

Definition 6.3.3 Let $\mathcal{C}$ be a class $\mathcal{C}$ of distributional decision problems. For a set $\mathcal{F}$ of distributions, set

$$\mathrm{Aver}(\mathbf{P},\mathcal{F})^{\mathcal{C}} = \{(D,\mu) \mid \exists (E,\nu) \in \mathcal{C}\ [(D,\mu) \in \mathrm{Aver}(\mathbf{P},\mathcal{F})^{(E,\nu)}]\}.$$

Propositions 5.2.11 and 5.5.7 immediately yield the following closure properties.

Proposition 6.3.4 **[97]** $\mathrm{Aver}(\mathbf{P},\mathcal{F})^{\mathrm{Aver}(\mathbf{P},*)} = \mathrm{Aver}(\mathbf{P},\mathcal{F})$ *for any set $\mathcal{F}$ of distributions.*

Proof. Clearly $\mathrm{Aver}(\mathbf{P},\mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P},\mathcal{F})^{\mathrm{Aver}(\mathbf{P},*)}$ since $(A,\mu) \in \mathrm{Aver}(\mathbf{P},\mathcal{F})^{(A,\mu)}$. For the other direction, consider any problem (A,μ) in $\mathrm{Aver}(\mathbf{P},\mathcal{F})^{\mathrm{Aver}(\mathbf{P},*)}$. There exists a problem $(B,\nu) \in \mathrm{Aver}(\mathbf{P},*)$ to which (A,μ) is avp-T-reducible. Using Theorem 5.2.12, we conclude that (A,μ) belongs to $\mathrm{Aver}(\mathbf{P},*)$. Since $\mu \in \mathcal{F}$, we get $(A,\mu) \in \mathrm{Aver}(\mathbf{P},\mathcal{F})$. □

6.3.2 Relativized Aver($\mathbf{BPP}, \mathcal{F}$)

We have seen the average-case version of bounded-error probabilistic reducibility in Section 5.5. Although the reducibility defined there is truth-table reducibility, we can easily extend it to Turing reducibility which will provide a relativization of the average-case complexity class Aver($\mathbf{BPP}, \mathcal{F}$) in this subsection.

Definition 6.3.5 (Relativization of Aver($\mathbf{BPP}, \mathcal{F}$)) Let (B, ν) be a distributional decision problem. For a set $\mathcal{F}$ of distributions, denote by Aver($\mathbf{BPP}, \mathcal{F}$)$^{(B,\nu)}$ the collection of problems (A, μ) such that there exist a probabilistic oracle Turing machine M, a real number ϵ in the interval $(0, 1/2)$, a distributional decision problem (B', ν'), and a semi-distribution η satisfying the following conditions:

(i) (Efficiency) M with oracle B' is polynomial-time bounded on μ-average;

(ii) (Validity) $\mathbf{Pr}_M[M^{B'}(x) = A(x)] \geq \frac{1}{2} + \epsilon$ for all x;

(iii) (Domination) $\mu \preceq^{\mathrm{avrp}}_{\Gamma_{M^{B'}}} \eta$ and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{(x, s) \mid z \in Q(M, B', x, s)\})$; and

(iv) (Transformation) $(B', \nu') \leq^{\mathrm{p}}_m (B, \nu)$.

If conditions (i)-(iii) are satisfied, we simply say that M *recognizes* A *with oracle* (B', ν') *in polynomial time on* μ*-average.*

The reader may pay attention to the fact that our relativization of Aver($\mathbf{BPP}, \mathcal{F}$) is a Turing extension of avbpp-tt-reducibility. Since avbpp-tt-reducibility is transitive (Proposition 5.5.7), $(A, \mu) \leq^{\mathrm{avbpp}}_{tt} (B, \nu)$ implies $(A, \mu) \in \mathrm{Aver}(\mathbf{BPP}, *)^{(B,\nu)}$.

We first discuss some of the basic properties of the relativized Aver($\mathbf{BPP}, \mathcal{F}$).

Proposition 6.3.6 *Let (A, μ) and (B, ν) be any distributional decision problems, and let $\mathcal{F}$ be any set of distributions.*

1. $(B, \nu) \in \mathrm{Aver}(\mathbf{BPP}, \mathcal{F})^{(B,\nu)}$.
2. *The class* Aver($\mathbf{BPP}, \mathcal{F}$)$^{(B,\nu)}$ *is closed under avbpp-tt-reductions.*
3. *If* $(A, \mu) \in \mathrm{Aver}(\mathbf{P}, *)^{(B,\nu)}$, *then* Aver($\mathbf{BPP}, \mathcal{F}$)$^{(A,\mu)} \subseteq$ Aver($\mathbf{BPP}, \mathcal{F}$)$^{(B,\nu)}$.

Proof. (1) By an argument similar to that of Proposition 6.3.2(1). (2) The claim follows from Proposition 5.5.7. (3) The claim follows by a an argument similar to Proposition 5.5.7. □

As in the definition of Aver($\mathbf{P}, \mathcal{F}$)$^{\mathcal{C}}$, we extend Definition 6.3.5 from a single oracle problem to a class $\mathcal{C}$ of oracle problems.

Definition 6.3.7 If $\mathcal{C}$ is a class of distributional decision problems and if $\mathcal{F}$ is a set of distributions, let

$$\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})^{\mathcal{C}} = \{(D, \mu) \mid \exists (E, \nu) \in \mathcal{C}\ [(D, \mu) \in \mathrm{Aver}(\mathbf{BPP}, \mathcal{F})^{(E,\nu)}]\}.$$

Next we show a collapse of $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$ relative to $\mathrm{Aver}(\mathbf{BPP}, *)$ down to $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$.

Proposition 6.3.8 $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{BPP},*)} = \mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$ *for any set* $\mathcal{F}$ *of distributions.*

Proof. To prove this proposition, we assume that (A, μ) is in $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})^{(B,\nu)}$ for some $(B, \nu) \in \mathrm{Aver}(\mathbf{BPP}, *)$. In what follows, we shall prove that (A, μ) belongs to $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$.

Let M be a bounded-error probabilistic Turing machine which recognizes A with oracle (B, ν) in polynomial time on μ-average. We can assume without loss of generality that the error probability of the machine M with oracle B on input x is bounded above by $1/4$.

Let $B' = \{z01^k \mid z \in B\}$ and let

$$\hat{\nu}'(w) = \begin{cases} \hat{\nu}(z) \cdot \hat{\nu}_{\mathrm{tally}}(1^k) & \text{if } w = z01^k, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $(B'\nu')$ is p-m-reducible to (B, ν). Since $\mathrm{Aver}(\mathbf{BPP}, *)$ is closed under p-m-reductions, (B', ν') is in $\mathrm{Aver}(\mathbf{BPP}, *)$. Using Corollary 3.5.32, let M_{amp} be a bounded-error probabilistic Turing machine which recognizes B' in polynomial time on ν'-average with the error probability $\mathbf{Pr}_s[M_{amp}(z01^k; s) \neq B'(z01^k)] \leq 2^{-k}$.

Let us define the randomized Turing machine M_0 as follows:

begin randomized algorithm for M_0 with an oracle
  input x
  simulate M on input x until the first query is made
  if there is no query **then** output $M(x)$ and **halt**
  for $k = 1$ **to** ∞ **do**
    resume the simulation of M on input x
      until the next query is made
(*) **if** M queries z **then** query $z01^{k+3}$ instead and
      receive an answer from oracle
    if M reaches a halting configuration **then output** $M(x)$ and **halt**
  end-for
end.

Note that the probability $\mathbf{Pr}_s[M_0^{B'}(x; s) = A(x)]$ is equal to the probability $\mathbf{Pr}_{s_k}[M^B(x; s_k) = A(x)]$, which is is at least $1/4$. Thus, it is easy to show that A is recognized by M_0 with oracle (B', ν') by an argument similar to that for Lemma 5.5.2.

Next we change the line (*) from making a query $z01^{k+3}$ to simulating M_{amp} on input $z01^{k+3}$. Let N be the randomized Turing machine obtained by this modification. By Lemma 5.5.3, it follows that the running time of the machine N is polynomial on μ-average.

Consider the error probability $e_N(x)$ of the machine N on input x. On each computation path generated by M_0 with oracle B' on input x, the error probability that N draws an erroneous conclusion is at most the

sum of all error probabilities which are made by the machine M_{amp}, which does not exceed:

$$2^{-4} + 2^{-5} + \cdots = \sum_{k=1}^{\infty} 2^{-k-3} = 2^{-4} \sum_{k=0}^{\infty} 2^{-k} = \frac{1}{8}.$$

Thus, we have:

$$e_M(x) \leq \mathbf{Pr}_s[M_0^{B'}(x;s) \neq A(x)] + \frac{1}{4} \leq \frac{1}{4} + \frac{1}{8} = \frac{3}{8}.$$

Hence, $e_N(x) \leq 3/8$.

Therefore, (A, μ) is in Aver$(\mathbf{BPP}, \mathcal{F})$. □

6.3.3 Relativized Aver$(\mathbf{NP}, \mathcal{F})$

This section introduces a relativization of Aver$(\mathbf{NP}, \mathcal{F})$ which will be used to build an "average polynomial-time hierarchy" in Section 6.4.

Schuler and Yamakami [97] first studied an average-case version of nondeterministic Turing reducibility and introduced a relativized class Aver$(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$ requiring that all computation paths be pruned to the same length, using a model of *clocked* nondeterministic oracle Turing machines. Based on this model, they introduced a "biased" domination condition in such a way that it puts equal weight on all accepting computation paths if one exists, or else puts equal weight on all rejecting computation paths. Our model of nondeterministic Turing machines is more general, and we cannot take the same approach towards the relativization of Aver$(\mathbf{NP}, \mathcal{F})$. How can we define a relativization of Aver$(\mathbf{NP}, \mathcal{F})$? Steve Cook is credited with the following idea.

Let us recall the model of nondeterministic Turing machines. Our definition of the running time of nondeterministic Turing machines depends only on a shortest accepting computation path whenever it exists. When we look at a computation tree of a nondeterministic Turing machine on a particular input, we are interested only in one shortest accepting computation path, provided that one exists. Our domination condition for the oracle machine needs a constraint only on the computation paths of interest.

Let M be a nondeterministic oracle Turing machine and let A be an oracle. Let us recall that Acc(M, A, x) (Rej(M, A, x), resp.) denotes the set of (codes of) all accepting (rejecting, resp.) computation paths of M with oracle A on input x. We introduce a "flip-flop" set Flip(M, A, x) as follows:

$$\mathrm{Flip}(M, A, x) = \begin{cases} \mathrm{Acc}(M, A, x) & \text{if } \mathrm{Acc}(M, A, x) \neq \emptyset, \\ \mathrm{Rej}(M, A, x) & \text{otherwise.} \end{cases}$$

By $\alpha_M^A(x)$, we denote the (code of the) minimal computation path of M with oracle A on input x if one exists, and otherwise, the (code of the) minimal rejecting computation path. Obviously $\alpha_M^A(x)$ is the minimal computation path in Flip(M, A, x).

Definition 6.3.9 (Relativized Aver$(\mathbf{NP}, \mathcal{F})$) Let (B, ν) be a distributional decision problem. A distributional problem (A, μ) is in Aver$(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$ if there exist a nondeterministic oracle Turing machine M and a semi-distribution η such that

(i) (Efficiency) M with oracle B is polynomial-time bounded on μ-average;

(ii) (Validity) $A = L(M, B)$; and

(iii) (Domination) $\mu \preceq^{\mathrm{avp}} \eta$, and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{x \mid z \in Q(M, B, x, \alpha_M^B(x))\})$.

As in Turing reducibility, we call condition (iii) the *domination condition for* M.

We begin with a technical lemma.

Lemma 6.3.10 *Let M be a nondeterministic oracle Turing machine, g a function from Σ^* to $\mathbb{R}^+$, and let μ, ν be two distributions. Assume that M witnesses $(D, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(E,\nu)}$ and let*

$$h(x) = \min_{y \in \mathrm{Flip}(M,E,x)} \sum_{z \in Q(M,E,x,y)} g(z).$$

If g is polynomial on ν-average, then h is polynomial on μ-average.

Proof. Assume that $(D, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(E,\nu)}$ via a nondeterministic oracle Turing machine M. There exist a semi-distribution η and a polynomial p_D such that $D = L(M, E)$, $\mu \preceq^{\mathrm{avp}} \eta$, $\lambda x.\mathrm{Time}_M^E(x)$ is p_D on μ-average, and $\hat{\nu}(z) \geq \hat{\eta}(\{x \mid z \in Q(M, E, x, \alpha_M^E(x))\})$ for all z. Without loss of generality, we assume that $\mathrm{Time}_M^E(x) > |x|$ for all x.

Choose a polynomial p_E such that g is p_E on ν-average. Moreover, let p be a polynomial and q a function such that q is p on μ-average and $q(x) \cdot \hat{\eta}(x) \geq \hat{\mu}(x)$ for all x. We can assume that all polynomials, p_D, p_E and p, are increasing.

Now define a polynomial s as

$$s(z) = p_D(3z) \cdot p_E\left(p_D(3z) \cdot 6z^2 \cdot p(3z)\right) + c_0,$$

where $c_0 = h(\lambda)$.

We shall show that h is s on μ-average. For simplicity, let A_x and R_x denote $\mathrm{Acc}(M, E, x)$ and $\mathrm{Rej}(M, E, x)$, respectively. Also let $Q_{x,y}$ denote $Q(M, E, x, y)$. Let $B_r = \{x \in \Sigma^+ \mid \mathrm{Time}_M^E(x) \leq p_D(|x| \cdot 3r) \wedge q(x) \leq p(|x| \cdot 3r)\}$. For any real number $r > 0$,

$$\begin{aligned}
&\hat{\mu}(\{x \mid h(x) > s(|x| \cdot r)\}) \\
&\qquad \leq \quad \hat{\mu}(\{x \mid \mathrm{Time}_M^E(x) > p_D(|x| \cdot 3r)\}) + \hat{\mu}(\{x \mid q(x) > p(|x| \cdot 3r)\}) \\
&\qquad\qquad + \hat{\mu}(\{x \in B_r \mid h(x) > s(|x| \cdot r)\}).
\end{aligned}$$

Clearly the first two terms are bounded above by $1/3r$. To complete the proof, we should show that the last term (say, T_r) is also bounded by $1/3r$.

Fix $x \in B_r$ and assume that $h(x) > s(|x| \cdot r)$.

Claim 15 *For every $x \in B_r$, $\exists z \in Q_{x,\alpha_M^E(x)}[g(z) > p_E(|z| \cdot 6r|x|^2 \cdot p(|x| \cdot 3r))]$.*

Proof of Claim. First we consider the case that $x \in D$. From the minimality of h, it follows that $\forall y \in A_x[\sum_{z \in Q_{x,y}} g(z) > s(|x| \cdot r)]$, and in consequence, $\forall y \in A_x \exists z \in Q_{x,y}[\|Q_{x,y}\| \cdot g(z) > s(|x| \cdot r)]$. Now take the minimal accepting computation path $\alpha_M^E(x)$ of M with E on input x. Then, for some z in $Q_{x,\alpha_M^E(x)}$, $\|Q_{x,\alpha_M^E(x)}\| \cdot g(z) > s(|x| \cdot r)$. For such a z,

$$|z| \leq \|Q_{x,\alpha_M^E(x)}\| \leq \mathrm{Time}_M^E(x) \leq p_D(|x| \cdot 3r),$$

and as a result, $g(z) > p_E(|z| \cdot 6(r|x|)^2 \cdot p(|x| \cdot 3r)) \geq p_E(|z| \cdot 6r|x|^2 \cdot p(|x| \cdot r))$, since p_E is increasing. Therefore, $\exists z \in Q_{x,\alpha_M^A(x)}[g(z) > p_E(|z| \cdot 6r|x|^2 \cdot p(|x| \cdot 3r))]$.

The other case $x \in \overline{D}$ follows in a similar fashion. ■

Note that, for any set A, $\hat{\nu}(\{z \mid z \in A\}) \geq \hat{\eta}(\{x \mid \exists z \in Q_{x,\alpha_M^E(x)}[z \in A]\})$. Hence, the bound on T_r is calculated further as follows:

$$\begin{aligned}
T_r &\leq \sum_{n=1}^{\infty} \hat{\mu}(\{x \in B_r \cap \Sigma^n \mid \exists z \in Q_{x,\alpha_M^E(x)}[g(z) > p_E(|z| \cdot 6rn^2 \cdot p(3rn))]\}) \\
&\leq \sum_{n=1}^{\infty} p(3rn) \cdot \hat{\eta}(\{x \in \Sigma^n \mid \exists z \in Q_{x,\alpha_M^E(x)}[g(z) > p_E(|z| \cdot 6rn^2 \cdot p(3rn))]\}) \\
&\leq \sum_{n=1}^{\infty} p(3rn) \cdot \hat{\nu}(\{z \mid g(z) > p_E(|z| \cdot 6rn^2 \cdot p(3rn))\}) \\
&\leq \sum_{n=1}^{\infty} p(3rn) \cdot \frac{1}{6rn^2 \cdot p(3rn)} \;=\; \frac{\pi^2}{36r} \;<\; \frac{1}{3r}.
\end{aligned}$$

Hence, we obtain the inequality $\hat{\mu}(\{x \mid h(x) > s(r \cdot |x|)\}) < 1/r$, and this implies that h is s on μ-average. □

The following is a list of basic properties which $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$ satisfies.

Proposition 6.3.11 *Let (A, μ) and (B, ν) be distributional decision problems, and let $\mathcal{F}$ be any set of distributions.*

1. $(B, \nu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$.
2. $\mathrm{Aver}(\mathbf{P}, \mathcal{F})^{(B,\nu)} \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$.
3. *The class* $\mathrm{Aver}(\mathbf{NP}, *)^{(B,\nu)}$ *is closed downward under avp-m-reductions.*
4. *If* $(A, \mu) \in \mathrm{Aver}(\mathbf{P}, *)^{(B,\nu)}$, *then* $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(A,\mu)} \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$.

Proof. (1)-(2) Clear from the definitions.

(3) The idea of the proof is similar to that of Proposition 5.5.7(5). Assume that $(A_1, \mu_1) \leq_m^{\mathrm{avp}} (A_2, \mu_2)$ via a reduction f such that $(f, \mu_1) \in \mathrm{Aver}(\mathbf{FP}, *)$. Assume that a nondeterministic Turing machine M computes A_2 with oracle (A_3, μ_3) in time p_M on μ_2-average, where p_M is a polynomial. Let M_f be a

deterministic transducer which computes f in time p_f on μ_1-average for some polynomial p_f. We note that $|f(x)| \leq \mathrm{Time}_{M_f}(x)$ for all x.

To improve the readability, we write $\alpha(x)$ instead of $\alpha_M^{A_3}(x)$. By the domination condition for (M, A_3), there exist a semi-distribution η_2 and a function p_M which is polynomial on μ_2-average such that $p_M(w) \cdot \hat{\eta}_2(w) \geq \hat{\mu}_2(w)$ and $\hat{\mu}_3(z) \geq \hat{\eta}_2(\{w \mid z \in Q(M, A_3, w, \alpha(w))\})$. Similarly, there are a semi-distribution η_1 and a function p_f which is q on μ_1-average, where q is a polynomial, such that $p_f(x) \cdot \hat{\eta}_1(x) \geq \hat{\mu}_1(x)$ and $\hat{\mu}_2(w) \geq \hat{\eta}_1(\{x \mid f(x) = w\})$. Without loss of generality, we assume that $p_M(x) \geq 2$ and $p_f(x) \geq 2$ for all strings x. Notice that

$$\hat{\mu}_2(w) \geq \hat{\eta}_1(\{x \mid f(x) = w\}) \geq \sum_{x: f(x)=w} \frac{\hat{\mu}_1(x)}{p_f(x)}.$$

Next we consider the most conservative algorithm N which computes A_1 using oracle A_3: namely, on input x, simulate $M_f(x)$ first and then simulate M nondeterministically on input $f(x)$. Clearly

$$\mathrm{Time}_N^{A_3}(x) \leq c \cdot (\mathrm{Time}_{M_f}(x) + \mathrm{Time}_M^{A_3}(f(x)) + 1)$$

for some positive constant c independent of x.

To simplify the description, denote by Q_z the set $\{w \mid z \in Q(M, A_3, w, \alpha(w))\}$. Then,

$$\hat{\mu}_3(z) \geq \hat{\eta}_2(\{w \mid w \in Q_z\}) \geq \hat{\eta}_2(Q_z \cap \mathrm{ran}(f)) \geq \sum_{w \in Q_z \cap \mathrm{ran}(f)} \frac{\hat{\mu}_2(w)}{p_M(w)}.$$

Now fix w in $Q_z \cap \mathrm{ran}(f)$. For this w,

$$\frac{\hat{\mu}_2(w)}{p_M(w)} \geq \frac{\hat{\eta}_1(\{x \mid f(x) = w\})}{p_M(w)} \geq \sum_{x: f(x)=w} \frac{\hat{\mu}_1(x)}{p_M(w) \cdot p_f(x)} = \sum_{x: f(x)=w} \frac{\hat{\mu}_1(x)}{p_M(f(x)) \cdot p_f(x)}.$$

Thus,

$$\hat{\mu}_3(z) \geq \sum_{w \in Q_z} \sum_{x \in f^{-1}(w)} \frac{\hat{\mu}_1(x)}{p_M(f(x)) \cdot p_f(x)}.$$

This sum is taken over all x such that $\exists w \in \mathrm{ran}(f)[x \in f^{-1}(w) \wedge z \in Q(M, A_3, w, \alpha(w))]$. This condition is logically equivalent to the condition $z \in Q(N, A_3, x, \alpha(x))$. We define $p(x) = p_M(f(x)) \cdot p_f(x)$, and then,

$$\hat{\mu}_3(z) \geq \sum_{x: z \in Q(N, A_3, x, \alpha(x))} \frac{\hat{\mu}_1(x)}{p_M(f(x)) \cdot p_f(x)} = \sum_{x: z \in Q(N, A_3, x, \alpha(x))} \frac{\hat{\mu}_1(x)}{p(x)}.$$

To complete the proof, we should show that $\lambda x.\mathrm{Time}_N^{A_3}(x)$ and p are polynomial on μ_1-average since, by Lemma 5.2.4, $\lambda x.\mathrm{Time}_M^{A_3}(f(x))$ turns out to be polynomial on μ_1-average. However, by Lemmas 3.3.13 and 3.3.12, we can conclude that $\lambda x.\mathrm{Time}_N^{A_3}(x)$ is polynomial on μ_1-average. Since the case for p is similar, we omit its proof.

(4) To show the claim, we assume that $(A, \mu) \in \mathrm{Dist}(\mathbf{P}, *)^{(B,\nu)}$, and take an arbitrary distributional problem (D, ξ) in $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(A,\mu)}$. There exists a nondeterministic oracle Turing machine M_D which computes D with oracle (E, ν) in polynomial time on μ-average. We shall show that (D, ξ) belongs to $\mathrm{Aver}(\mathbf{NP}, *)^{(B,\nu)}$. Since $(A, \mu) \in \mathrm{Aver}(\mathbf{P}, *)^{(B,\nu)}$, we can find a deterministic oracle Turing machine M_A which computes A with oracle (B, ν) in polynomial time on μ-average.

Now we consider a machine M which nondeterministically simulates the computation of M_D, and whenever M_D makes a query z, M deterministically simulates M_A on input z. By definition, M computes D with the help of oracle B, and the running time of M with oracle B on input x is bounded above by

$$c \cdot \left(\mathrm{Time}^A_{M_D}(x) + \min_{y \in \mathrm{Flip}(M_D, A, x)} \sum_{z \in Q(M_D, A, x, y)} \mathrm{Time}^B_{M_A}(z) + 1 \right),$$

where c is a constant. Note that $\lambda x.\mathrm{Time}^A_{M_D}(x)$ is polynomial on ξ-average, and the function $\lambda x.\, \min_{y \in \mathrm{Flip}(M_D, A, x)} \sum_{z \in Q(M_D, E, x, y)} \mathrm{Time}^B_{M_E}(x)$ is also polynomial on ξ-average by Lemma 6.3.10. As a result, $\lambda x.\mathrm{Time}_M(x)$ is polynomial on ξ-average.

The proof of the domination condition for M is similar to that of Proposition 5.2.11(4). This yields the desired consequence that $(D, \xi) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$. □

We can conjecture that $\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp})$ differs from $\mathrm{Aver}(\mathbf{NP}, \mathbf{P}\text{-comp})$. As yet it remains an open question whether this is the case in the *unrelativized world*. In some relativized world, however, we can see a clear distinction between $\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp})$ and $\mathrm{Aver}(\mathbf{NP}, \mathbf{P}\text{-comp})$. Here we observe a relativized world in which the two classes differ.

Theorem 6.3.12 *There exists a problem (B, μ_B) such that*

$$\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp})^{(B,\mu_B)} \neq \mathrm{Aver}(\mathbf{NP}, \mathbf{P}\text{-comp})^{(B,\mu_B)}.$$

Proof. We shall use the oracle set constructed by Baker, Gill, and Solovay [3] to separate $\mathbf{P}$ from $\mathbf{NP}$. In the following proof, we first review their construction and then define the desired distributional decision problem (B, μ_B).

Also let $\{M_i\}_{n \in \mathbb{N}}$ be an effective enumeration of all polynomial-time deterministic oracle Turing machines. Let also $\{p_n\}_{n \in \mathbb{N}}$ be an enumeration of polynomials such that each p_n satisfies $\mathrm{Time}^O_{M_n}(x) \leq p_n(|x|)$ for any choice of oracle O and any input x. In particular, the number of query strings of M_n on input x with oracle O is also bounded above by $p_n(|x|)$.

Let us define the strictly increasing function ℓ from $\mathbb{N}$ to $\mathbb{N}$ as follows: let $\ell(-1) = 0$ for the sake of convenience, and let $\ell(n)$ be the minimal integer k such that $k > \ell(n-1)$ and $p_n(k) < 2^k$. It is easy to see that such a k exists for any n.

We use the test language $T(B)$ defined as $T(B) = \{0^n \mid \exists y \in \Sigma^n [y \in B]\}$. This $T(B)$ is a tally set and belongs to $\mathbf{NP}^B$. We shall construct a set B in the following such that $T(B) \notin \mathbf{P}^B$. First we construct a series of finite sets, $\{B_n\}_{n \in \mathbb{N}}$, and then let B be the union of all sets B_n. Now let $B_{-1} = \emptyset$ for the sake of convenience. For each $n \in \mathbb{N}$, let y_n be the minimal string (in the standard order on Σ^*) such that

$$|y_n| = \ell(n) \text{ and } y_n \notin Q(M_n, B_{n-1}, 0^{\ell(n)}).$$

Such a string exists because $\|Q(M_n, B_{n-1}0^{\ell(n)})\| \leq p_n(\ell(n)) < 2^{\ell(n)}$ by the definition of $\ell(n)$. Then, B_n is

defined as

$$B_n = \begin{cases} B_{n-1} \cup \{y_n\} & \text{if } M_n^{B_{n-1}}(0^{\ell(n)}) = 0, \\ B_{n-1} & \text{otherwise.} \end{cases}$$

Notice that the outcome of M_n on $0^{\ell(n)}$ is irrelevant to oracle B_n, i.e., $M_n^{B_{n-1}}(0^{\ell(n)}) = M_n^{B_n}(0^{\ell(n)})$, because y_n is not queried by M_n on $0^{\ell(n)}$. Hence, we get

$$0^{\ell(n)} \in T(B) \Longleftrightarrow y_n \in B_n \Longleftrightarrow M_n^{B_n}(0^{\ell(n)}) = 0 \Longleftrightarrow M_n^{B}(0^{\ell(n)}) = 0 \Longleftrightarrow 0^{\ell(n)} \notin L(M_n, B).$$

This shows that $T(B) \neq L(M_n, B)$ for all numbers n, and thus $T(B) \notin \mathbf{P}^B$.

We then define the distribution μ_B for the set B. Let

$$\hat{\mu}_B(x) \propto \begin{cases} (|x|+1)^{-2} & \text{if } x \in B, \\ (|x|+1)^{-2} \cdot 2^{-|x|} & \text{otherwise.} \end{cases}$$

Consider the distributional decision problem $(T(B), \nu_{\mathrm{tally}})$. Clearly $(T(B), \nu_{\mathrm{tally}})$ is in $\mathrm{Aver}(\mathbf{NP}, \mathbf{P}\text{-comp})^{(B,\mu_B)}$. Now assume to the contrary that $(T(B), \nu_{\mathrm{tally}})$ belongs to $\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp})^{(B,\mu_B)}$. There exists a deterministic Turing machine M which computes $T(B)$ with oracle (B, μ_B) in polynomial time on ν_{tally}-average. Recall that $T(B)$ is a tally set. Since M is polynomial-time bounded on ν_{tally}-average, we conclude that $T(B) \in \mathbf{P}^B$. This contradicts the fact that $T(B) \notin \mathbf{P}^B$. □

Finally we extend Definition 6.3.1 from a single oracle problem to a class of oracle problems.

Definition 6.3.13 (Relativization) Let $\mathcal{C}$ be a class of distributional decision problems and let $\mathcal{F}$ be a set of distributions. Let $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathcal{C}}$ denote the union of $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(E,\nu)}$ for any oracle (E, ν) chosen from $\mathcal{C}$.

As an immediate consequence of Proposition 6.3.11, the relativized $\mathrm{Aver}(\mathbf{NP}, *)$ relative to $\mathrm{Aver}(\mathbf{P}, *)$ collapses to $\mathrm{Aver}(\mathbf{NP}, *)$. Note that whether $\mathrm{Aver}(\mathbf{NP}, *)$ relative to $\mathrm{Aver}(\mathbf{NP}, *)$ collapses to $\mathrm{Aver}(\mathbf{NP}, *)$ is an open question.

Proposition 6.3.14 $\mathrm{Aver}(\mathbf{NP}, *) = \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{P}, *)}$.

Proposition 6.3.14 will be extended to any level of the average polynomial-time hierarchy in Section 6.4.

6.3.4 Relativized $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$

Another important average complexity class is the collection of deterministic average polynomial-space computable sets, $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$. This class contains $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$ and $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$ as subclasses. In this section, we shall introduce a relativization of $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$.

Definition 6.3.15 (Relativized $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$**)** Let (B, ν) be a distributional decision problem. A distributional problem (A, μ) is in $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(B,\nu)}$ if there exist a deterministic oracle Turing machine M and a semi-distribution η such that

(i) (Efficiency) M with oracle B is polynomial-space bounded on μ-average;

(ii) (Validity) $A = L(M, B)$; and

(iii) (Domination) $\mu \preceq^{\mathrm{avp}} \eta$, and $\hat{\nu} \geq \lambda z.\hat{\eta}(\{x \mid z \in Q(M, B, x)\})$.

We also call the condition (iii) the *domination condition for* M.

Since our relativization is similar to that of $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$, the following proposition is straightforward.

Proposition 6.3.16 *Let (A, μ) and (B, ν) be distributional decision problems.*

1. $(A, \mu) \in \mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(A,\mu)}$.
2. $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})^{(A,\mu)} \subseteq \mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(A,\mu)}$.
3. *If* $(A, \mu) \in \mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(B,\nu)}$, *then* $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(A,\mu)} \subseteq \mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(B,\nu)}$.

Theorem 6.3.17 *There exists a problem (A, μ_A) such that*

$$\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp})^{(A,\mu_A)} = \mathrm{Aver}(\mathbf{PSPACE}, \mathbf{P}\text{-comp})^{(A,\mu_A)}.$$

Proof. For the proof, we need a relativized version of the randomized bounded halting problem $(\mathrm{BHP}_1(A), \mu_{\mathrm{BHP}_1})$. Assume that $\{M_i\}_{i\in\mathbb{N}}$ is an effective enumeration of all deterministic oracle Turing machines. For a set A, let

$$\mathrm{BHP}_1(A) = \{\langle 0, s_i, x, 1^n\rangle \mid M_i \text{ with oracle } A \text{ accepts } x \text{ using less than } n \text{ squares }\}$$
$$\cup\{\langle 1, s_i, x, 1^n\rangle \mid M_i \text{ with oracle } A \text{ on } x \text{ uses at least } n \text{ squares }\};$$

and, for $b \in \{0, 1\}$, let

$$\hat{\mu}_{\mathrm{BHP}_1}(b, s_i, x, 1^n) = \frac{1}{2} \cdot \hat{\nu}_{\mathrm{stand}}(s_i) \cdot \hat{\nu}_{\mathrm{stand}}(x) \cdot \hat{\nu}_{\mathrm{tally}}(1^n).$$

Notice that the distribution μ_{BHP_1} is not dependent on oracle A.

Take the desired set A so that $A = \mathrm{BHP}_1(A)$. We remark that this set A exists because M_i cannot query any strings of length more than n, and thus, it makes only queries to oracle A that are lexicographically smaller than $\langle b, s_i, x, 1^n\rangle$. Now let $\mu_A = \mu_{\mathrm{BHP}_1}$. We shall show that any distributional problem (D, μ) in $\mathrm{Aver}(\mathbf{PSPACE}, \mathbf{P}\text{-comp})^{(A,\mu_A)}$ belongs to $\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp})^{(A,\mu_A)}$. For this pair (D, μ), there exists a deterministic Turing machine M computing D with oracle A in polynomial time on μ-average.

Let g be the function of Lemma 4.2.7(2) such that $\hat{\mu}(x) < 2^{-|g(x)|+2}$ for all x, and define the machine M' as follows: on input x, it computes $g(x)$ and then simulates M on input $g(x)$. Let i be an index such that $M_i = M'$ since M' is also deterministic. Let us consider the following deterministic procedure N:

begin deterministic algorithm N
  input x
  compute $g^{-1}(x)$ and set $u := g^{-1}(x)$
  for $n = 1$ **to** ∞ **do**
    query $\langle 0, s_i, u, 1^n \rangle$ to oracle A
    if $\langle 0, s_i, u, 1^n \rangle \in A$ **then** accept and **halt**
    query $\langle 1, s_i, u, 1^n \rangle$ to oracle A
    if $\langle 1, s_i, u, 1^n \rangle \notin A$ **then** reject and **halt**
  end-for
end.

Notice that the machine N makes queries only of the form $\langle b, s_i, g^{-1}(x), 1^n \rangle$, where $b \in \{0,1\}$ and $1 \leq n \leq \mathrm{Time}^A_{M_i}(x) + 1$.

It is not difficult to check that, with oracle A, N computes D correctly. We next check the domination condition for N. Remember that, for each query string $\langle b, s_i, u, 1^n \rangle$, the string $g(u)$ is the only input on which N queries it. Since g is one-one, this correspondence from query strings to inputs is also one-one. Thus, the rest of the proof is analogous to the proof of Claim 11 in Section 5.3.5.

Let $q(x) = 512 \cdot (|g(x)|+1)^2 \cdot (\mathrm{Time}^A_{M_i}(x)+1)^2 / \hat{\nu}_{\mathrm{stand}}(s_i)$. It is obvious that q is polynomial on μ-average because $\lambda x.\mathrm{Time}^A_{M_i}(x)$ is polynomial on μ-average. As in Theorem 5.3.2, we have

$$\begin{aligned} q(x) \cdot \hat{\mu}_A(b, s_i, g(x), 1^n) &= q(x) \cdot \frac{1}{2} \cdot \hat{\nu}_{\mathrm{stand}}(s_i) \cdot \hat{\nu}_{\mathrm{stand}}(x) \cdot \hat{\nu}_{\mathrm{tally}}(1^n) \\ &\geq \frac{q(x) \cdot \hat{\nu}_{\mathrm{stand}}(s_i)}{512 \cdot (|g(x)|+1)^2 \cdot (\mathrm{Time}^A_{M_i}(x)+1)^2} \cdot 2^{-|g(x)|+2} \\ &= 2^{-|g(x)|+2} \;>\; \hat{\mu}(x). \end{aligned}$$

This indicates that (D, μ) belongs to $\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp})^{(D,\mu)}$. □

We then introduce a relativized class $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{\mathcal{C}}$ for a class $\mathcal{C}$ of distributional decision problems.

Definition 6.3.18 (Relativization) Let $\mathcal{C}$ be a class of distributional decision problems. For a set $\mathcal{F}$ of distributions, $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{\mathcal{C}}$ ($\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$ relative to $\mathcal{C}$) denotes the collection of all distributional problems in $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(B,\nu)}$ for some $(B, \nu) \in \mathcal{C}$.

The class $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$ relative to $\mathrm{Aver}(\mathbf{PSPACE}, *)$ collapses to $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$.

Proposition 6.3.19 *For any set $\mathcal{F}$ of distributions,*

$$\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F}) = \mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{\mathrm{Aver}(\mathbf{PSPACE}, *)}.$$

Proof. It is clear that $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{\mathrm{Aver}(\mathbf{PSPACE}, *)}$. We shall show the other inclusion. Assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})^{(B,\nu)}$ for some $(B, \nu) \in \mathrm{Aver}(\mathbf{PSPACE}, *)$. Let M be a deterministic oracle Turing machine which computes A with oracle (B, ν) using polynomial space on μ-average. Since $(B, \nu) \in \mathrm{Aver}(\mathbf{PSPACE}, *)$, there exists a deterministic Turing machine M_B which computes B using polynomial space on ν-average. The algorithm we would like to consider here is the conservative one: on input x, simulate M on x except for oracle queries; instead of querying a string z, simulate M_B on z. On each input x, this algorithm uses space at most

$$c \cdot \left(\mathrm{Space}_M^B(x) + \max_{z \in Q(M,B,x)} \mathrm{Space}_{M_B}(z) + 1 \right)$$

for some fixed constant $c > 0$.

We can conclude that this bound is polynomial on μ-average since $\lambda x. \max_{z \in Q(M,B,x)} \mathrm{Space}_M^B(z)$ is polynomial on μ-average as in Lemma 6.3.10. Therefore, (A, μ) belongs to $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$. □

6.4 Average Polynomial-Time Hierarchy

This section will formally introduce an average-case version of the polynomial-time hierarchy. In previous sections, we introduced relativized $\mathrm{Aver}(\mathbf{P}, \mathcal{F})$ and relativized $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$. These relativized classes will be the basis for constructing an average-case version of the polynomial-time hierarchy.

6.4.1 Average Polynomial Time Hierarchy

We now give a formal definition of an *average polynomial-time hierarchy* under a particular set of distributions. This hierarchy is an average-case analogue of the polynomial-time hierarchy in worst-case complexity theory.

Definition 6.4.1 (Average Polynomial-Time Hierarchy) [97] Let $k > 1$ and let $\mathcal{F}$ be a set of distributions.

1. $\mathrm{Aver}(\mathbf{\Delta}_0^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{\Sigma}_0^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{P}, \mathcal{F})$.
2. $\mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_{k-1}^{\mathrm{p}}, *)}$.
3. $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_{k-1}^{\mathrm{p}}, *)}$.
4. $\mathrm{Aver}(\mathbf{\Pi}_k^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\text{co-}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$.
5. $\mathrm{Aver}(\mathbf{PH}, \mathcal{F}) = \bigcup_{k>0} \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$.

In what follows, we shall show several fundamental properties of the average polynomial-time hierarchy.

Lemma 6.4.2 **[97]** *Let $k > 0$ and let $\mathcal{F}$ be a set of distributions.*

1. *The classes* $\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})$ *and* $\mathrm{Aver}(\mathbf{\Pi}^{\mathrm{p}}_k, \mathcal{F})$ *are closed under avp-m-reductions.*

2. *The class* $\mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, \mathcal{F})$ *is closed under avp-T-reductions.*

Proof. (1) In the case $k = 1$, the claim for $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$ follows from Proposition 5.2.6. Now let $k \geq 2$ and assume that $(A, \mu) \leq^{\mathrm{avp}}_m (B, \nu)$ and $(B, \nu) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})$. By definition, there exists a problem $(C, \xi) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k-1}, *)$ such that $(B, \nu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(C,\xi)}$. Proposition 6.3.11(3) implies that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(C,\xi)}$. Using the definition again, we obtain $(A, \mu) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})$.

For the class $\mathrm{Aver}(\mathbf{\Pi}^{\mathrm{p}}_k, \mathcal{F})$, assume that $(A, \mu) \leq^{\mathrm{avp}}_m (B, \nu)$ and also (B, ν) is in $\mathrm{Aver}(\mathbf{\Pi}^{\mathrm{p}}_k, \mathcal{F})$. Note that $(A, \mu) \leq^{\mathrm{avp}}_m (B, \nu)$ if and only if $(\overline{A}, \mu) \leq^{\mathrm{avp}}_m (\overline{B}, \nu)$. From this fact, it follows that $(\overline{A}, \mu) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})$. This is equivalent to $(A, \mu) \in \mathrm{Aver}(\mathbf{\Pi}^{\mathrm{p}}_k, \mathcal{F})$.

(2) Similar to (1). If $k = 1$, then the claim is based on Proposition 5.2.6. For the other case, we use Proposition 5.5.7(5). □

In what follows, we shall see the basic inclusions among the classes of the average polynomial-time hierarchy. First we want to see the lemma that characterizes the relationship between $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})$ and $\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})$.

Lemma 6.4.3 *Let $k > 0$ and let $\mathcal{F}$ be a set of distributions.*

1. $\mathrm{Dist}(\mathbf{\Delta}^{\mathrm{p}}_{k+1}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)}$.

2. $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_{k+1}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)}$.

Proof. We shall show only case (2) because case (1) follows by a similar argument. Let us assume that (A, μ) is in $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_{k+1}, \mathcal{F})$. Since $A \in \mathbf{\Sigma}^{\mathrm{p}}_{k+1}$, there exists a nondeterministic oracle Turing machine M which computes A in polynomial time with oracle B in $\mathbf{\Sigma}^{\mathrm{p}}_k$. By a simple modification of M and B, we can assume the following property: on each input x, the machine makes a query once of the form $\langle x, y \rangle$, where y is the (code of the) computation path. Note that, for every query string, there is the unique pair of an input and a path.

We shall show that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$ for some ν. Let us first define such a distribution. Let ν be defined as:

$$\hat{\nu}(z) = \begin{cases} \hat{\mu}(x) & \text{if } z = \langle x, \alpha^B_M(x) \rangle, \\ 0 & \text{otherwise.} \end{cases}$$

Our definition of ν obviously guarantees the domination condition for M. Hence, M recognizes A with oracle B in polynomial time. Since $(B, \nu) \in \mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)$, we obtain:

$$(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)} \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)}.$$

□

Using Lemma 6.4.3, we can prove the following proposition.

Proposition 6.4.4 **[97]** *Let $k \geq 1$ and let $\mathcal{F}$ be any set of distributions.*

1. $\mathrm{Dist}(\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F})$.
2. $\mathrm{Dist}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$.
3. $\mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{\Delta}_{k+1}^{\mathrm{p}}, \mathcal{F})$.

Proof. (1)-(2) The claims follow from Lemma 6.4.3. (3) This claim follows from Definition 6.4.1 and from the facts that $\mathrm{Aver}(\mathbf{P}, \mathcal{F})^{\mathcal{C}} \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathcal{C}}$ and $(D, \mu) \in \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{(D,\mu)}$. □

The above proposition may be taken as evidence that our average polynomial-time hierarchy has a structure similar to that of the worst-case polynomial-time hierarchy.

Figure 6.1 illustrates the structure of the average polynomial-time hierarchy under $\mathcal{F}$.

Lemma 6.4.5 *Let $k > 0$. For a set $\mathcal{F}$ of distributions,* $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Delta}_{k+1}^{\mathrm{p}}, *)} = \mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, \mathcal{F})$.

Proof. Assume that (A, μ) is in $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Delta}_{k+1}^{\mathrm{p}}, *)}$. By definition, we then have a chain of membership relations $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$ and $(B, \nu) \in \mathrm{Aver}(\mathbf{P}, *)^{(C,\xi)}$, where $(C, \xi) \in \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$. By Proposition 6.3.11(4), we can shorten this chain to $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(C,\xi)}$. This shows that (A, μ) is in $\mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, \mathcal{F})$.

The converse is even simpler to prove. Since $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *) \subseteq \mathrm{Aver}(\mathbf{\Delta}_{k+1}^{\mathrm{p}}, *)$, it follows that

$$\mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)} \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Delta}_{k+1}^{\mathrm{p}}, *)}.$$

□

Theorem 3.5.24 shows the separation between $\mathrm{Aver}(\mathbf{P}, *)$ and $\mathrm{Aver}(\mathbf{NP}, *)$. Although these basic average-case complexity classes are different, we do not conclude that similar separations occur in the higher levels of the average polynomial-time hierarchy. The reason is that domination conditions restrict the complexity of oracles, especially distributions.

The worst-case polynomial-time hierarchy entails the "downward collapse" property; that is, if any two levels of the hierarchy collapse, then the upper levels collapse down to those levels. More precisely, if $\mathbf{\Sigma}_k^{\mathrm{p}} = \mathbf{\Sigma}_{k+1}^{\mathrm{p}}$, then $\mathbf{PH} = \mathbf{\Sigma}_k^{\mathrm{p}}$. Now we show that the assumption $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *) = \mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, *)$ leads to the collapse of the average polynomial-time hierarchy.

Proposition 6.4.6 *Let $k \geq 1$ and let $\mathcal{F}$ be a set of distributions.*

1. *If* $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *) = \mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, *)$, *then* $\mathrm{Aver}(\mathbf{PH}, \mathcal{F}) = \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$.

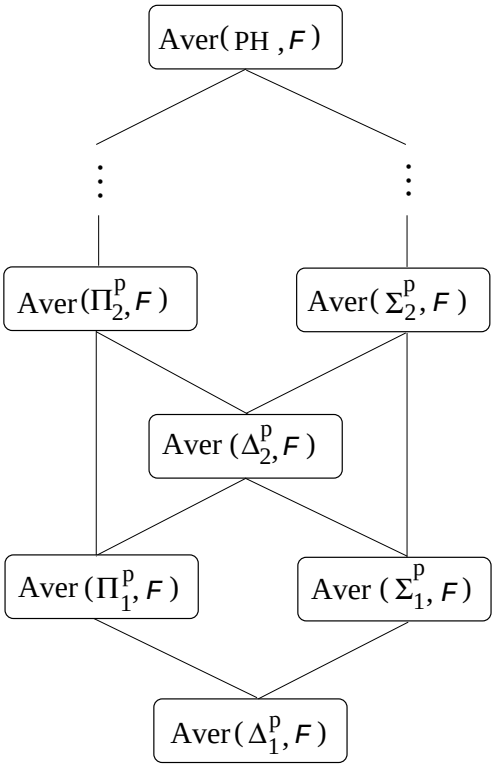

Figure 6.1: The average polynomial-time hierarchy under $\mathcal{F}$

2. *If* $\mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, *) = \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$, *then* $\mathrm{Aver}(\mathbf{PH}, \mathcal{F}) = \mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F})$.

Proof. (1) Let us assume that $\mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, *)$ collapses to $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$. We want to show by induction on integer $i \geq k$ that $\mathrm{Aver}(\mathbf{\Sigma}_{i+1}^{\mathrm{p}}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{\Sigma}_i^{\mathrm{p}}, \mathcal{F})$ for all sets $\mathcal{F}$ of distributions.

The base case $i = k$ is trivial from our assumption. For the induction step $i > k$, we have

$$\begin{aligned} \mathrm{Aver}(\mathbf{\Sigma}_{i+2}^{\mathrm{p}}, \mathcal{F}) &= \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_{i+1}^{\mathrm{p}}, *)} \\ &\subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_{i}^{\mathrm{p}}, *)} \\ &= \mathrm{Aver}(\mathbf{\Sigma}_{i+1}^{\mathrm{p}}, \mathcal{F}). \end{aligned}$$

Hence, $\mathrm{Aver}(\mathbf{\Sigma}_{k+i}^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$ for all $i \geq 0$. Therefore, $\mathrm{Aver}(\mathbf{PH}, \mathcal{F}) = \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$.

(2) Similarly, we have:

$$\begin{aligned} \mathrm{Aver}(\mathbf{\Sigma}_{k+2}^{\mathrm{p}}, \mathcal{F}) &= \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, *)} \\ &= \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Delta}_{k+1}^{\mathrm{p}}, *)} \\ &= \mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, \mathcal{F}). \end{aligned}$$

The last equality follows from Lemma 6.4.5. □

The following lemma is an extension of Lemma 3.5.16.

Lemma 6.4.7 *Let $\mathcal{C} \in \{\mathbf{\Delta}_k^{\mathrm{p}}, \mathbf{\Sigma}_k^{\mathrm{p}}, \mathbf{\Pi}_k^{\mathrm{p}} \mid k > 0\}$. Assume that $\mathcal{F}$ is closed under $\oplus$. If (A, μ_A) and (B, μ_B) are both in $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$, then so is $(A \oplus B, \mu_A \oplus \mu_B)$.*

Proof. Assume that (A, μ_A) and (B, μ_B) are in $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$. The proof is by induction on $k \geq 1$. For the base case $k = 1$, the claim for $\mathcal{C} \in \{\mathbf{P}, \mathbf{NP}\}$ is immediate from Lemma 3.5.16. Now let us consider the case $\mathcal{C} = \text{co-}\mathbf{NP}$. Assume that (A, μ_A) and (B, μ_B) are in $\mathrm{Aver}(\text{co-}\mathbf{NP}, \mathcal{F})$. In other words, $(\overline{A}, \mu_A)$ and $(\overline{B}, \mu_B)$ are in $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$. Again by Lemma 3.5.16, we have $(\overline{A} \oplus \overline{B}, \mu_A \oplus \mu_B) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})$. Notice that $\overline{A} \oplus \overline{B} = \overline{A \oplus B} - \{\lambda\}$, where λ is the empty string. Since $\widehat{\mu_A \oplus \mu_B}(\lambda) = 0$, we may ignore λ, and thus $(\overline{A \oplus B}, \mu_A \oplus \mu_B) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})$. This yields the desired conclusion that $(A \oplus B, \mu_A \oplus \mu_B) \in \mathrm{Aver}(\text{co-}\mathbf{NP}, \mathcal{F})$.

For the induction step $k > 1$, first let $\mathcal{C} = \mathbf{\Sigma}_k^{\mathrm{p}}$. By our assumption, there exist distributional problems (C_0, ξ_0) and (C_1, ξ_1), both of which belongs to $\mathrm{Aver}(\mathbf{\Sigma}_{k-1}^{\mathrm{p}}, \mathcal{F})$, such that $(A, \mu_A) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(C_0, \xi_0)}$ and $(B, \mu_B) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(C_1, \xi_1)}$. It is relatively easy to see that $(C_0, \xi_0) \leq_m^{\mathrm{p}} (C_0 \oplus C_1, \xi_0 \oplus \mu_1)$ and $(C_1, \xi_1) \leq_m^{\mathrm{p}} (C_0 \oplus C_1, \xi_0 \oplus \mu_1)$. For simplicity, write (C, ξ) for $(C_0 \oplus C_1, \xi_0 \oplus \mu_1)$. By our induction hypothesis, we obtain $(C, \xi) \in \mathrm{Aver}(\mathbf{\Sigma}_{k-1}^{\mathrm{p}}, \mathcal{F})$. Using Lemma 7.2.12, we get $(A \oplus B, \mu_A \oplus \mu_B) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(C, \xi)}$. As (C, ξ) is in $\mathrm{Aver}(\mathbf{\Sigma}_{k-1}^{\mathrm{p}}, \mathcal{F})$, the problem $(A \oplus B, \mu_A \oplus \mu_B)$ is in $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$. The case $\mathcal{C} = \mathbf{\Delta}_k^{\mathrm{p}}$ is similar, and the other case $\mathcal{C} = \mathbf{\Pi}_k^{\mathrm{p}}$ is obtained with the same idea used for the base case $\mathcal{C} = \text{co-}\mathbf{NP}$. □

At last, we may conjecture that, for a naturally selected set $\mathcal{F}$ of distributions, the average polynomial-time hierarchy under $\mathcal{F}$ is truly an infinite hierarchy.

6.4.2 Sparse Interpolation Property

We shall show a basic relationship between worst-case complexity and average-case complexity on strings with high probability. We first introduce an *interpolation* property of an average-case complexity class $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$. Intuitively, the property says that if we compute a set A fast on average under a distribution which assigns high probability to instances in a sparse set S, then there is an "interpolant" set B between $A \cap S$ and A which is computable fast in worst-case. This set B is a collection of "easy" instances in A, and it becomes a good worst-case approximation of the set A.

Here is the formal definition of the *sparse interpolation property*.

Definition 6.4.8 (Sparse Interpolation Property) [97] For a sparse set S and a polynomial q, let $\mu_{S,q}$ denote a distribution such that

$$\hat{\mu}_{S,q}(x) \geq \frac{1}{q(|x|)}$$

for all $x \in S$ provided that $\sum_{x \in S} \frac{1}{q(|x|)} \leq 1$. A class $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ has the *sparse interpolation property* if, for any set A, any infinite sparse set S, and any polynomial q such that $(A, \mu_{S,q}) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$, there exists a set $B \in \mathcal{C}$ such that $A \cap S \subseteq B \subseteq A$. The set B is called an *interpolant* of A and S.

Proposition 6.4.9 *For a class $\mathcal{C} \in \{\mathbf{P}, \mathbf{NP}, \mathbf{RP}, \mathbf{BPP}, \mathbf{PSPACE}\}$, $\mathrm{Aver}(\mathcal{C}, *)$ has the sparse interpolation property.*

Proof. We first show the case $\mathcal{C} = \mathbf{NP}$. Take any sparse set S and a polynomial q and assume that $(A, \mu_{S,q}) \in \mathrm{Aver}(\mathbf{NP}, *)$. There exists a nondeterministic Turing machine M which computes A such that $\lambda x.\mathrm{Time}_M(x)$ is p on $\mu_{S,q}$-average for some polynomial p. Note that $\mathrm{Time}_M(x) \leq p(|x|/\hat{\mu}_{S,q}(x))$ for all x with $\hat{\mu}_{S,q}(x) > 0$. Let N simulate M on input x in $p(|x| \cdot q(|x|))$ steps. If the simulation of M does not terminate within $p(|x| \cdot q(|x|))$ steps, then N rejects x. Let $B = L(N)$. Clearly $B \subseteq A$. Since $q(|x|) \geq 1/\hat{\mu}_{S,q}(x)$ for all $x \in S$, N completely simulates M on all inputs x in S. Thus, $A \cap S = B \cap S$. Clearly N is polynomial-time bounded. Therefore, $B \in \mathbf{NP}$.

The other cases are treated similarly, but specifically for the case $\mathcal{C} \in \{\mathbf{RP}, \mathbf{BPP}\}$, we must use $\mathrm{Time}^*_M(x)$ instead of $\mathrm{Time}_M(x; s)$. □

Proposition 6.4.9 can be extended to an arbitrary level of the average polynomial-time hierarchy. However, its proof is not as simple as that of Proposition 6.4.9. We first present a key lemma.

Lemma 6.4.10 **[97]** *Let $k \geq 1$ and let (A, μ) be a distributional problem.*

1. Assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, *)$. For any set S and any polynomial q, there exist sets $C_0, C_1 \in \mathbf{\Delta}^{\mathrm{p}}_k$ and S' such that $S' \subseteq S$, $A \cap S' \subseteq C_0 \subseteq A$, $\overline{A} \cap S' \subseteq C_1 \subseteq \overline{A}$ and $\hat{\mu}(S^n - S'^n) \leq 1/q(n)$ for all $n \in \mathbb{N}$.

2. Assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)$. For any set S and any polynomial q, there exist sets $C_0 \in \mathbf{\Sigma}^{\mathrm{p}}_k$, $C_1 \in \mathbf{\Pi}^{\mathrm{p}}_k$ and S' such that $S' \subseteq S$, $A \cap S' \subseteq C_0 \subseteq A$, $\overline{A} \cap S' \subseteq C_1 \subseteq \overline{A}$ and $\hat{\mu}(S^n - S'^n) \leq 1/q(n)$ for all $n \in \mathbb{N}$.

Proof. (1) The proof proceeds by induction on k. The base case $k = 1$ essentially follows from Proposition 6.4.9. Assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, *)$ and let $S \subseteq \Sigma^*$ and let q be any polynomial. There are a nondeterministic Turing machine M and a polynomial p such that $A = L(M)$ and $\hat{\mu}(\{x \mid \mathrm{Time}_M(x) > p(|x| \cdot r)\}) < 1/r$ for all $r > 0$.

Let $r = q(n)$, and then $\hat{\mu}(\{x \mid \mathrm{Time}M(x) > p(|x| \cdot q(n))\}) < 1/q(n)$. Let M_0 simulate M on the same input in time $p(n \cdot q(n))$; if M accepts x, then M_0 accepts it, or else M_0 rejects x. Similarly, let M_1 simulate M in time $p(n \cdot q(n))$; if M rejects x, then M_1 accepts it, or else M_1 rejects it. Define $C_0 = L(M_0)$ and $C_1 = L(M_1)$, and let $S' = S \cap (C_0 \cup C_1)$. Clearly $A \cap S' \subseteq C_0 \subseteq A$ and $\overline{A} \cap S' \subseteq C_1 \subseteq \overline{A}$. Moreover,

$$\hat{\mu}(S^n - S'^n) \leq \hat{\mu}(\{x \mid \mathrm{Time}_M(x) > p(|x| \cdot q(n))\}) < \frac{1}{q(n)}.$$

The induction step is carried out as follows. Let $k \geq 2$ and assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)$. By definition, there exists a distributional problem $(B, \nu) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k-1}, *)$ such that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B,\nu)}$. Let M be a nondeterministic oracle Turing machine which computes A with oracle (B, ν) in polynomial time on μ-average. We assume that $\lambda x.\mathrm{Time}^B_M(x)$ is p on μ-average for some increasing polynomial p. The domination condition for M implies the existence of a semi-distribution η and a function d which is

polynomial on μ-average such that $d(x) \cdot \hat{\eta}(x) \geq \hat{\eta}(x)$ and $\hat{\nu}(z) \geq \hat{\eta}(\{x \mid z \in Q(M, B, x, \alpha_M^B(x))\})$ for all x and z. Assume also that t is a polynomial witnessing that d is polynomial on μ-average.

Consider any set S and any polynomial q. We define T as

$$T = \{x \in S \mid \mathrm{Time}_M^B(x) \leq p(|x| \cdot 3q(|x|)) \wedge d(x) \leq t(|x| \cdot 3q(|x|))\}.$$

For each $n \in \mathbb{N}$, let us consider the subset T^n of T. Notice that

$$S^n - T^n \subseteq \{x \in \Sigma^n \mid \mathrm{Time}_M^B(x) > p(|x| \cdot 3q(n)) \vee d(x) \leq t(|x| \cdot 3q(n))\}.$$

Then, it follows that

$$\hat{\mu}(S^n) - \hat{\mu}(T^n) = \hat{\mu}(S^n - T^n) \leq \hat{\mu}(\{x \mid \mathrm{Time}_M^B(x) > p(|x| \cdot 3q(n))\}) + \hat{\mu}(\{x \mid d(x) \leq t(|x| \cdot 3q(n))\} \leq \frac{2}{3q(n)}.$$

In the rest of the proof, we use $s(n)$ for $t(n \cdot 3q(n))$.

To use the induction hypothesis, we let $Z = \{z \mid \exists x \in T(z \in Q(M, B, x, \alpha_M^B(x)))\}$, and we also let

$$l(n) = 3q(n) \cdot s(n) \cdot p(n \cdot 3q(n)).$$

Applying the induction hypothesis to Z and $l(n)$, it follows that there exist sets $Z' \subseteq Z$, $C_0' \in \mathbf{\Sigma}_{k-1}^{\mathrm{p}}$, and $C_1' \in \mathbf{\Pi}_{k-1}^{\mathrm{p}}$ such that, for every $n \in \mathbb{N}$, $\hat{\nu}(Z^n) - \hat{\nu}(Z'^n) \leq 1/l(n)$.

Now recall that M queries strings whose length is at least the same as that of the input and at most the size of its running time. In other words, for every x in T^n, if $z \in Q(M, B, x, \alpha_M^B(x))$, then $n \leq |z| \leq p(n \cdot 3q(n))$. Based on this fact, we prepare sets $\tilde{Z}_n = \{z \in Z \mid n \leq |z| \leq p(n \cdot 3q(n))\}$ and $\tilde{Z}_n' = \{z \in Z' \mid n \leq |z| \leq p(n \cdot 3q(n))\}$. Since

$$\tilde{Z}_n - \tilde{Z}_n' \subseteq (Z - Z') \cap \{z \mid n \leq |z| \leq p(n \cdot 3q(n))\},$$

it follows that

$$\hat{\nu}(\tilde{Z}_n) - \hat{\nu}(\tilde{Z}_n') \leq \sum_{i=n}^{p(n \cdot 3q(n))} (\hat{\nu}(Z^i) - \hat{\nu}(Z'^i)) \leq \sum_{i=n}^{p(n \cdot 3q(n))} \frac{1}{l(i)} \leq \frac{p(n \cdot 3q(n))}{l(n)} = \frac{1}{3q(n) \cdot s(n)}.$$

The desired set S' is defined as follows:

$$S' = \{x \in T \mid Q(M, B, x, \alpha_M^B(x)) \subseteq Z'\}.$$

Clearly $S' \subseteq S$. Using the fact that

$$T^n - S'^n \subseteq \{x \in T^n \mid Q(M, B, x, \alpha_M^B(x)) \cap (\tilde{Z}_n - \tilde{Z}_n') \neq \emptyset\},$$

by the domination condition, we obtain $\hat{\nu}(\tilde{Z}_n - \tilde{Z}_n') \geq \hat{\eta}(T^n - S'^n)$. Then,

$$\hat{\mu}(T^n) - \hat{\mu}(S'^n) = \hat{\mu}(T^n - S'^n) \leq s(n) \cdot \hat{\eta}(T^n - S'^n) \leq s(n) \cdot \hat{\nu}(\tilde{Z}_n - \tilde{Z}_n') \leq \frac{1}{3q(n)}.$$

Since $\hat{\mu}(S^n) - \hat{\mu}(T^n) \leq \frac{2}{3q(n)}$, we get $\hat{\mu}(S^n) - \hat{\mu}(S'^n) \leq \frac{1}{q(n)}$.

Let M_0 be an oracle Turing machine with oracle X defined as follows. On input x, M_0 simulates M on x in time $p(|x| \cdot 3q(|x|))$, and whenever M queries a string z, M_0 queries both $0z$ and $1z$ to its oracle X. If

$0z \in X$ and $1z \notin X$, M_0 continues the simulation with the assumption that the oracle answer is "yes"; if $0z \notin X$ and $1z \in X$, then it continues the simulation with the oracle answer "no"; otherwise, it immediately rejects the input x. The machine M_0 accepts x exactly when M halts and accepts it. Similarly, we define a machine M_1 by interchanging the oracle answers and requiring that M_1 accept the input x if M halts in time $p(|x| \cdot 3q(|x|))$ and otherwise M_1 rejects x. Now let $C_0 = L(M_0, C_0' \oplus C_1')$ and $C_1 = L(M_1, C_0' \oplus C_1')$. By the definitions of the oracle machines M_0 and M_1, it follows that $A \cap S' \subseteq C_0 \subseteq A$ and $\overline{A} \cap S' \subseteq C_1 \subseteq \overline{A}$.

(2) The proof is similar to (1). □

Theorem 6.4.11 **[97]** *For $k \geq 1$, $\mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, *)$ and $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$ have the sparse interpolation property.*

Proof. We show only the case $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$ here. The case $k = 1$ follows from Proposition 6.4.9. Let $k \geq 2$ and assume that $(A, \mu_{S,q}) \in \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$ for a sparse set S and a polynomial q. It follows from Lemma 6.4.10 that there exist a set $C \in \mathbf{\Sigma}_k^{\mathrm{p}}$ and a subset S' of S such that $A \cap S' \subseteq C \subseteq A$ and $\hat{\mu}_{S,q}(S^n) - \hat{\mu}_{S,q}(S'^n) \leq 1/2q(n)$ for all $n \in \mathbb{N}$. It suffices to show that $S' = S$. Assume that there exists a string $x \in S - S'$. Let $n = |x|$. Since $\hat{\mu}_{S,q}(x) \geq 1/q(|x|)$, it follows that

$$\frac{1}{q(n)} \leq \hat{\mu}_{S,q}(S^n) - \hat{\mu}_{S,q}(S'^n) \leq \frac{1}{2q(n)}.$$

This is a contradiction. Hence, $S' = S$. □

The following proposition is another consequence of Lemma 6.4.10, and it will be used in the next chapter.

Proposition 6.4.12 *Let $k > 0$. For any polynomial p and any set A such that $(A, \nu_{\mathrm{stand}}) \in \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$, there are two sets C_0 in $\mathbf{\Sigma}_k^{\mathrm{p}}$ and C_1 in $\mathbf{\Pi}_k^{\mathrm{p}}$ such that $C_0 \subseteq A$, $C_1 \subseteq \overline{A}$, $\|C_0^n \cup C_1^n\| \geq (1 - 1/p(n)) \cdot 2^n$ for almost all n.*

Proof. Let p be any polynomial. Assume $(A, \nu_{\mathrm{stand}}) \in \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$. Take $q(n) = 2(n+1)^2 \cdot p(n)$. Note that

$$\frac{2^{2\mathrm{llog}(n)+1}}{q(n)} \leq \frac{2(n+1)^2}{2(n+1)^2 \cdot p(n)} = \frac{1}{p(n)}.$$

We apply Lemma 6.4.10 to q, A, and Σ^n. Then, we have sets $C_0 \in \Sigma_k$, $C_1 \in \mathbf{\Pi}_k^{\mathrm{p}}$, and S' such that $A \cap S' \subseteq C_0 \subseteq A$, $\overline{A} \cap S' \subseteq C_1 \subseteq \overline{A}$, and $\hat{\nu}_{\mathrm{stand}}(\Sigma^n - S'^n) \leq 1/q(n)$ for all $n \in \mathbb{N}$. Obviously, $S' \subseteq C_0 \cup C_1$. Hence, $\|C_0^n \cup C_1^n\| \geq \|S'^n\|$. It suffices to show that $\|S'n\| \geq (1 - \frac{1}{p(n)}) \cdot 2^n$. Note that

$$\hat{\nu}_{\mathrm{stand}}(\Sigma^n - S') = \hat{\nu}_{\mathrm{stand}}(\Sigma^n) - \hat{\nu}_{\mathrm{stand}}(S'^n) = 2^{-2\mathrm{llog}(n)-1} - \|S'^n\| \cdot 2^{-n-2\mathrm{llog}(n)-1}.$$

Since $\hat{\nu}_{\mathrm{stand}}(\Sigma^n - S'n) \leq 1/q(n)$,

$$\begin{aligned} \|S'^n\| &\geq \left(2^{-2\mathrm{llog}(n)-1} - \frac{1}{q(n)}\right) \cdot 2^{n+2\mathrm{llog}(n)+1} \\ &\geq \left(1 - \frac{2^{2\mathrm{llog}(n)+1}}{q(n)}\right) \cdot 2^n \;\geq\; \left(1 - \frac{1}{p(n)}\right) \cdot 2^n. \end{aligned}$$

□

6.5 Average Polynomial-Time Alternation Hierarchy

In worst-case complexity theory, alternation plays a variety of roles. Recall that $\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}} = \mathrm{ATIME}^{\Sigma}(k, n^{O(1)})$ and $\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}} = \mathrm{ATIME}^{\Delta}(k, n^{O(1)})$. Then, we have $\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}} = \mathbf{\Sigma}_k^{\mathrm{p}}$ and $\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}} = \mathbf{\Delta}_k^{\mathrm{p}}$ for all $k > 0$; namely, the polynomial-time alternation hierarchy is exactly the polynomial-time hierarchy (see Lemma 2.5.7). In what follows, we introduce an average version of this polynomial-time alternation hierarchy, an average polynomial-time alternation hierarchy under a set of distributions, and study its relationship to the average polynomial-time hierarchy defined in Section 6.4.

Definition 6.5.1 (Average Polynomial Time Alternation Hierarchy under $\mathcal{F}$) Let $k > 0$ and let $\mathcal{F}$ be a set of distributions. The *average polynomial-time alternating hierarchy under* $\mathcal{F}$ consists of the following average classes:

1. $\mathrm{Aver}(\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}, \mathcal{F}) = \bigcup_{c>0} \mathrm{Aver}(\mathrm{ATIME}^{\Delta}(k, n^c + c), \mathcal{F})$.
2. $\mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F}) = \bigcup_{c>0} \mathrm{Aver}(\mathrm{ATIME}^{\Sigma}(k, n^c + c), \mathcal{F})$.
3. $\mathrm{Aver}(\mathbf{A}\mathbf{\Pi}_k^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\text{co-}\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$.

Let $\mathrm{Aver}(\mathbf{APH}, \mathcal{F}) = \bigcup_{k>0} \mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$.

Recall that 1-alternation bounded alternating Turing machines with existential states are exactly the nondeterministic Turing machines. Similarly, semi-deterministic 1-alternation bounded alternating Turing machines are deterministic Turing machines. Hence, it is clear that $\mathrm{Aver}(\mathbf{A}\mathbf{\Delta}_1^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{P}, \mathcal{F})$ and $\mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}_1^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{NP}, \mathcal{F})$ for all set $\mathcal{F}$.

Nevertheless, two classes $\mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$ and $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$ are unlikely to be the same even for the set $\mathcal{F}$ of feasibly computable distributions. The following proposition helps us understand the gap between these two classes. Recall that $\mathrm{Aver}(\mathbf{\Delta}_{k+1}^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)}$ and $\mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)}$.

Theorem 6.5.2 *Let $k > 0$ and $\mathcal{F}$ be a set of distributions.*

1. $\mathrm{Aver}(\mathbf{A}\mathbf{\Delta}_{k+1}^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{\mathrm{Dist}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)}$.
2. $\mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, \mathcal{F}) = \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Dist}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)}$.

Proof. We prove (2) here. First we assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, \mathcal{F})$. There exists a $(k+1)$-alternation bounded Turing machine M such that $A = L(M)$ and $\lambda x.\mathrm{Time}_M(x)$ is p on μ-average, where p is a polynomial. We shall show that A is computable by a nondeterministic oracle Turing machine N with oracle (B, ν) in polynomial time on μ-average.

The main idea here is to query sufficiently long strings to the oracle so that the complexity of the oracle set can be substantially diminished. We begin by defining the oracle Turing machine N as follows:

begin nondeterministic algorithm for N
  input x (assuming that $x \neq \lambda$)
  simulate M on input x during the first series of existential states
    until M reaches the universal states
  (let y be the code of such a computation path)
  choose a path starting from y nondeterministically (ignoring the states of configurations)
    and follow this path until M reaches a halting configuration
  (let n be the length of the path from the initial configuration)
  query the pair $\langle x, y10^n \rangle$ to the oracle
  if oracle answers "yes" **then** reject **else** accept
end.

Clearly N is a nondeterministic Turing machine whose running time is $O(\mathrm{Time}_M(x))$.

Let B be the set that is accepted by the following algorithm. On input $\langle x, y10^n \rangle$, check if y encodes a computation of the first series of existential states of M on input x, and if so, then simulate this path until M goes into universal states. Simulate the rest of the computation of M within n steps. If a path does not halt within n steps, then go into an accepting configuration immediately. It is not difficult to see that $\overline{B}$ is in $\mathrm{ATIME}^{\Sigma}(k, O(n))$, and thus $\overline{B} \in \mathbf{\Sigma}^{\mathrm{p}}_k$. By definition, $A = L(N, \overline{B})$. To see that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(\overline{B}, \nu)}$ for some ν, we remark first that $Q(N, \overline{B}, x, \alpha^{\overline{B}}_M(x)) = \{\langle x, y10^n \rangle\}$ for some y and n. Next we define

$$\hat{\nu}(z) = \begin{cases} \hat{\mu}(\{x \mid z \in Q(N, \overline{B}, x, \alpha^{\overline{B}}_N(x))\}) & \text{if } z \neq \lambda, \\ 1 - \sum_{w : w \neq \lambda} \hat{\nu}(w) & \text{if } z = \lambda. \end{cases}$$

It is easy to check that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, *)^{(\overline{B}, \nu)}$ via N. Since $(\overline{B}, \nu)$ is in $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)$, we have $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)}$.

Conversely, assume that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)}$. There exists a problem $(B, \nu) \in \mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)$ such that $(A, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(B, \nu)}$ via an appropriate nondeterministic oracle Turing machine M. Assume that B is in $\mathrm{ATIME}^{\Sigma}(k, p(n))$ for some polynomial p. We can assume that p is strictly increasing.

Next we define an alternating Turing machine N as follows. On input x, N simulates M on input x starting with an existential state. During this existential state, if M queries z, then N stores z (in a work tape) and guesses its oracle answer $ans(z)$ and continue the simulation until M terminates. On each computation path, if M reaches an accepting configuration, then N existentially checks if $z_i \in B$ for all z_i with $ans(z_i) = 1$. Then N universally checks if $z_i \notin B$ for all z_i which satisfy $ans(z_i) = 0$ at once using B. If $\chi_B(z_i) = 1$ for all such z_i, then accept x; otherwise, reject x.

It is easy to check that $A = L(N)$. The running time of N on input x is at most

$$\mathrm{Time}_N(x) \leq c \cdot (\mathrm{Time}_M(x) \cdot p(\mathrm{Time}_M(x)) + 1)$$

for some constant $c > 0$. Since $\lambda x.\mathrm{Time}_M(x)$ is polynomial on μ-average, by Lemma 3.3.14, $\lambda x.\mathrm{Time}_N(x)$ is also polynomial on μ-average. We then have $(A,\mu) \in \mathrm{Aver}(\mathrm{ATIME}^{\Sigma}(k+1, n^{O(1)}), \mathcal{F}) = \mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{k+1}, \mathcal{F})$. □

Corollary 6.5.3 *Let $k > 0$ and $\mathcal{F}$ be a set of distributions.*

1. $\mathrm{Dist}(\mathbf{\Delta}^{\mathrm{p}}_k, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_k, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, \mathcal{F})$.

2. $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})$.

In worst-case complexity theory, there is a nice characterization of the class **PSPACE** by polynomial-time alternating Turing machines: that is, $\mathbf{PSPACE} = \mathrm{ATIME}(n^{O(1)})$. Interestingly, we do not know whether $\mathrm{Aver}(\mathbf{PSPACE}, \mathcal{F})$ equals $\mathrm{Aver}(\mathrm{ATIME}(n^{O(1)}), \mathcal{F})$ because of the definition of the running time of alternating Turing machines.

In what follows, we shall show that $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$ for a supportive set $\mathcal{F}$ of distributions is located within the average polynomial-time alternation hierarchy under $\mathcal{F}$.

Proposition 6.5.4 *For any supportive set $\mathcal{F}$ of distributions,* $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_2 \cap \mathbf{A}\mathbf{\Pi}^{\mathrm{p}}_2, \mathcal{F})$.

Proof. Assume that $\mathcal{F}$ is supportive and let (D,μ) be an arbitrary problem in $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$. Since μ is supportive, we can define a strictly increasing, positive polynomial p such that $\hat{\mu}(x) \geq 2^{-p(|x|)}$ for all x. By Proposition 3.5.33, there exists a randomized Turing machine M which recognizes D in polynomial time on μ-average with error probability $2^{-p(|x|)^2}$, i.e., $\mathbf{Pr}_M[M(x) \neq D(x)] \leq 2^{-p(|x|)^2}$.

Now let us define

$$\mathrm{Time}^*_M(x) = \begin{cases} \min\{n \mid \mathbf{Pr}_s[\mathrm{Time}_M(x;s) \leq n \mid s \in \Omega_M(x)] > \frac{1}{2}\} & \text{if one exists,} \\ \infty & \text{otherwise.} \end{cases}$$

Apply Lemma 3.3.21 to $\lambda x.\mathrm{Time}^*_M(x)$, and we conclude that $\lambda x.\mathrm{Time}^*_M(x)$ is polynomial on μ-average. Let q be a polynomial such that $\lambda x.\mathrm{Time}^*_M(x)$ is q on μ-average. For this q, we have $\mathrm{Time}^*_M(x) \leq q(|x|/\hat{\mu}(x))$ unless $\hat{\mu}(x) = 0$. Let $S = \{x \mid \hat{\mu}(x) > 0\}$. Then, for almost all x in S, $\mathrm{Time}^*_M(x) \leq q(|x| \cdot 2^{p(|x|)}) < 2^{p(|x|)^2 - 2}$. We take a positive integer n_0 satisfying $\log(\mathrm{Time}^*_M(x)) < p(|x|)^2 - 2$ for all $x \in S$ with $|x| \geq n_0$.

We next claim that the probability $\rho_x = \mathbf{Pr}_s[M(x;s) = D(x) \mid \mathrm{Time}_M(x;s) \leq \mathrm{Time}^*_M(x) \wedge s \in \Omega_M(x)]$ is at least $1 - 2^{p(|x|)^2+1}$. First we have:

$$\begin{aligned} &\mathbf{Pr}_s[M(x;s) = D(x) \wedge \mathrm{Time}_M(x;s) \leq \mathrm{Time}^*_M(x) \mid s \in \Omega_M(x)] \\ &\quad \geq 1 - \mathbf{Pr}_s[M(x;s) \neq D(x) \mid s \in \Omega_M(x)] - \mathbf{Pr}_s[\mathrm{Time}_M(x;s) \leq \mathrm{Time}^*_M(x) \mid s \in \Omega_M(x)] \\ &\quad \geq 1 - 2^{-p(|x|)^2} - \frac{1}{2} \;=\; \frac{1}{2}(1 - 2^{-p(|x|)^2+1}). \end{aligned}$$

Notice by the definition of $\mathrm{Time}^*_M(x)$ that $\mathbf{Pr}_s[\mathrm{Time}_M(x;s) \leq \mathrm{Time}^*_M(x) \mid s \in \Omega_M(x)] \geq 1/2$. Hence, the conditional success probability ρ_x is at least

$$\rho_x \geq \frac{\frac{1}{2}(1 - 2^{-p(|x|)^2+1})}{\frac{1}{2}} = 1 - 2^{-p(|x|)^2+1}.$$

In particular, when $m = \mathrm{Time}^*_M(x)$, $\rho_x \geq 1 - 2^{-p(|x|)^2+1} \geq 1 - 2^{-\log m - 1}$ since $\log m < p(|x|)^2 - 2$.

Let us define another Turing machine M' that simulates M by using an additional input string as a random seed. Formally, the algorithm is as follows:

begin deterministic algorithm for M'
  input (x, y) (say $y = y_1 y_2 \cdots y_m$, where $y_i \in \{0, 1\}$, $1 \leq i \leq m$)
  let $i := 0$
  start the simulation of M on input x
  while the simulation **do**
    if M flips a coin and $i \leq m$ **then** let its outcome be y_i
    if M flips a coin and $i > m$ **then** enter an infinite loop
  let $i := i + 1$
  end-while
end.

Note that we do not require the machine M' to exhaust all bits of y on each computation path of M on x; thus, M' halts on input (x, y) exactly when s is a prefix of y for some random seed $s \in \Gamma_M(x)$.

Towards achieving our goal, it suffices from Proposition 6.5.2(2) to show that (D, μ) belongs to $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(E,\nu)}$ for some distributional problem (E, ν) in $\mathrm{Dist}(\mathbf{NP}, *)$.

We first define a nondeterministic oracle Turing machine M_0 as follows:

begin nondeterministic algorithm M_0 with an oracle
  input x
  if $|x| < n_0$ **then output** $D(x)$
  guess a string w
  let $m = |w|$
  simulate M' on input (x, w) for m steps
  if either M' does not halt within m steps or M' enters a rejecting state
      then reject
  guess m distinct strings $u_1, \ldots, u_m$ of length m
  query the string $\langle x, u_1 \cdots u_m, w \rangle$ to oracle
  if the string is *not* in the oracle **then accept else reject**
end.

We now show that, for any oracle, M_0 is polynomial-time bounded on μ-average. To show this, it suffices to consider the case where $|w| = \mathrm{Time}^*_M(x)$. First notice that, for any oracle O, $\mathrm{Time}^O_{M_0}(x) \leq c \cdot (m +$

$m^2+1)$, where c is an appropriate constant. Since $\lambda x.\mathrm{Time}^*_M(x)$ is polynomial on μ-average, the function $\lambda x.\mathrm{Time}^O_{M_0}(x)$ is also polynomial on μ-average.

Next we shall define the desired oracle E so that $D = L(M_0, E)$. The set E is defined as the set computed by the nondeterministic machine M_1 below. Let $u \oplus v$ denote the bitwise addition of u and v modulo 2, and let the algorithm for M_1 be as follows:

begin nondeterministic algorithm for M_1
  input $\langle x, u_1 \cdots u_m, w\rangle$
  if $|u_1 \cdots u_m| \neq |w|^2$ **then accept**
  (Now assume that $m = |w|$.)
  guess a string v of length m
  for $i = 1$ **to** m **do**
    simulate M' on input $(x, u_i \oplus v)$ for m steps
    if M' does not halt **then accept**
    simulate M' on input (x, w) m steps
    if M' does not halt **then accept**
    if $M'(x, u_i \oplus v) \neq M'(x, w)$ **then accept else reject**
  end-for
end.

Note that M_1 is polynomial-time bounded, and as a result, E belongs to **NP**. Still we must prove that $D = L(M_0, E)$. For simplicity, we fix x and set $m = \mathrm{Time}^*_M(x)$. Let $A = \{(w, y) \in \Sigma^m \times \Sigma^m \mid M'(x, w) = M'(x, y)\}$. Moreover, let

$$B = \{w \in \Sigma^m \mid \text{ there are more than } 2^{m-\log m-1} \text{ strings } y \text{ of length } m \text{ such that } (w, y) \in A\}.$$

We now claim that:

Claim 16 *$w \in B$ if and only if $\exists(u_1, \ldots, u_m) \in (\Sigma^m)^m \exists v \in \Sigma^m \forall i \leq m[(w, u_i \oplus v) \in A]$.*

Proof of Claim. (Only if – part) Assume that w is in B. To produce a contradiction, we further assume that the right side of the above equivalence is false; namely, for every m-tuple $(u_1, \ldots, u_m)$ in $(\Sigma^m)^m$, there exists a string $v \in \Sigma^m$ such that $(w, u_i \oplus v) \notin A$ for all i, $1 \leq i \leq m$. Let $\{v_0, v_1, \ldots, v_{2^m-1}\}$ be an enumeration of all strings of length m. For each j, we define U_j as the set $\{(u_1, \ldots, u_m) \in (\Sigma^m)^m \mid \forall i \leq m[(u_i \oplus v_j) \notin A]\}$. Define $U = \bigcup_{j=0}^{2^m-1} U_j$. Since $\|U\| = \|\Sigma^m\|^m = 2^{m^2}$, there exists a natural number j_0 such that $\|U_{j_0}\| \geq \|U\|/2^m$. This implies that $\|U_{j_0}\| \geq 2^{m^2-m}$.

Let $C = \{u \in \Sigma^m \mid (w, u) \notin A\}$. Towards a contradiction, we must show that $\|C\| > 2^{m-\log m-1}$. Notice that $\|C\|$ is equal to the cardinality of the set $\{u \in \Sigma^m \mid (w, u \oplus v_{j_0}) \notin A\}$ because of the operator $\oplus$. Given any m-tuple $(u_1, \ldots, u_m) \in U_{j_0}$,

$$\|C_{j_0}\|^m = \|\{u \in \Sigma^m \mid (w, u \oplus v_{j_0}) \notin A\}\|^m \geq \|U_{j_0}\|.$$

The cardinality of the set C_{j_0} thus is bounded above by

$$\|C_{j_0}\| \geq (2^{m^2-m})^{1/m} \geq 2^{m-1} > 2^{m-\log m-1}.$$

(If – part) Assume that the right side of the equivalence in the claim is true; namely, there exists a m-tuple $(u_1, \ldots, u_m) \in (\Sigma^m)^m$ such that, for every $v \in \Sigma^m$, $(w, u_i \oplus v) \in A$ holds for some i. Fix such an m-tuple $(u_1, \ldots, u_m)$. Let $C' = \{u \in \Sigma^m \mid (w, u) \in A\}$. For each $y \in C'$, there exists a number i such that $(u_i \oplus v) \in A$. For each i, let $C'_i = \{y \in \Sigma^m \mid \exists y[y = u_i \oplus v \wedge (w, y) \in A]\}$. Because of the definition of $\oplus$, $\|C'_i\| = \|C'_j\|$ for all pairs (i, j), $1 \leq i, j \leq m$. Since $\Sigma^m = \bigcup_{i=1}^m C'_i$, we have $\|C'_i\| \geq 2^m/m > 2^{m-log m-1}$. Therefore, w is in B. ■

Recall that $x \in D$ if and only if there exist more than $2^{m-\log(m)-1}$ strings y of length m such that $(w, y) \in A$. This is equivalent to saying that there exists a string w of length m such that $w \in B$ and $M'(x, w) = 1$. By the above claim, it holds that

$$x \in D \Longleftrightarrow \exists (u_1, \ldots, u_m) \in (\Sigma^m)^m [(x, u_1 \cdots u_m, w) \notin E].$$

This yields the equation $D = L(M_0, E)$.

Finally we define the desired distribution ν on E as follows. Using the function $\alpha^E_{M_0}(x)$, we set $\hat{\nu}(z) = \hat{\mu}(\{x \mid z \in Q(M_0, E, x, \alpha^E_{M_0}(x))\})$. It is straightforward to see that (D, μ) is in $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(E,\nu)}$, and consequently (D, μ) is in $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{(E,\nu)} \subseteq \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Dist}(\mathbf{NP},*)} = \mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_2, \mathcal{F})$.

The other claim that $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{A}\mathbf{\Pi}^{\mathrm{p}}_2, \mathcal{F})$ follows from the inclusions:

$$\mathrm{Aver}(\mathbf{BPP}, \mathcal{F}) = \text{co-}\mathrm{Aver}(\mathbf{BPP}, \mathcal{F}) \subseteq \text{co-}\mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_2, \mathcal{F}) = \mathrm{Aver}(\mathbf{A}\mathbf{\Pi}^{\mathrm{p}}_2, \mathcal{F}).$$

□

6.6 Average Low Hierarchy

The average polynomial-time hierarchy allows us to construct an average-case version of the *low hierarchy in* **NP** to refine the structure within $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$. Perhaps some **NP**-complete problems with natural distributions which are unknown to be either in $\mathrm{Aver}(\mathbf{P}, *)$ or p-m-complete for $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp})$ fall into a *low hierarchy in* $\mathrm{Aver}(\mathbf{NP}, \mathbf{P}\text{-comp})$.

We first define the relativized average polynomial-time hierarchy.

Definition 6.6.1 (Relativized Average Polynomial Time Hierarchy) Let $\mathcal{F}$ be a set of distributions. For a distributional decision problem (B, ν), the *relativized average polynomial-time hierarchy under* $\mathcal{F}$ *relative to* (B, ν) consists of the following classes:

1. $\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})^{(B,\nu)} = \mathrm{Aver}(\mathbf{NP}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k-1}, *)^{(B,\nu)}}$.
2. $\mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, \mathcal{F})^{(B,\nu)} = \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k-1}, *)^{(B,\nu)}}$.

3. $\mathrm{Aver}(\mathbf{\Pi}^{\mathrm{p}}_k, \mathcal{F})^{(B,\nu)} = \text{co-}\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})^{(B,\nu)}$.

Based on the relativized hierarchy $\{\mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, \mathcal{F})^{(E,\nu)}, \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, \mathcal{F})^{(E,\nu)} \mid k > 0\}$, we introduce the low hierarchy within the class $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$.

Definition 6.6.2 (Average Low Hierarchy under $\mathcal{F}$) Let $k \in \mathbb{N}$.

1. $\mathrm{Aver}(\mathbf{L\Delta}^{\mathrm{p}}_k, \mathcal{F}) = \{(D,\mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F}) \mid \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, *)^{(D,\mu)} \subseteq \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, *)\}$.

2. $\mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_k, \mathcal{F}) = \{(D,\mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F}) \mid \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)^{(D,\mu)} \subseteq \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)\}$.

3. $\mathrm{Aver}(\mathbf{L\Pi}^{\mathrm{p}}_k, \mathcal{F}) = \text{co-}\mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_k, \mathcal{F})$.

4. $\mathrm{Aver}(\mathbf{LPH}, \mathcal{F}) = \bigcup_{i\in\mathbb{N}} \mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_i, \mathcal{F})$.

Here we remark that it is open whether each class of the average low hierarchy enjoys the sparse interpolation property.

Lemma 6.6.3 *Let $k > 0$ and $\mathcal{F}$ be a set of distributions.*

1. $\mathrm{Aver}(\mathbf{L\Delta}^{\mathrm{p}}_1, \mathcal{F}) = \mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_0, \mathcal{F}) = \mathrm{Aver}(\mathbf{P}, \mathcal{F})$.

2. $\mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_k, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{L\Delta}^{\mathrm{p}}_{k+1}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_{k+1}, \mathcal{F})$.

Proof. (1) Since $\mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_1, *)^{(D,\mu)} = \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_0, *)^{(D,\mu)} = \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{(D,\mu)}$ by definition, we obtain the equality $\mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_0, \mathcal{F}) = \mathrm{Aver}(\mathbf{L\Delta}^{\mathrm{p}}_1, \mathcal{F})$. It is also clear that $\mathrm{Aver}(\mathbf{L\Delta}^{\mathrm{p}}_1, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P}, \mathcal{F})$ because $(D,\mu) \in \mathrm{Aver}(\mathbf{P}, *)^{(D,\mu)}$.

The other direction $\mathrm{Aver}(\mathbf{P}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{L\Delta}^{\mathrm{p}}_1, \mathcal{F})$ follows from the fact that $\mathrm{Aver}(\mathbf{P}, \mathcal{F}) = \mathrm{Aver}(\mathbf{P}, \mathcal{F})^{\mathrm{Aver}(\mathbf{P},*)}$, as shown in Lemma 6.3.4(1).

(2) For the first inclusion, let (D,μ) be any distributional problem in $\mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_k, \mathcal{F})$. By definition, we have $\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)^{(D,\mu)} \subseteq \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)$. We then obtain:

$$\mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_{k+1}, *)^{(D,\mu)} = \mathrm{Aver}(\mathbf{P}, *)^{\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k,*)^{(D,\mu)}} \subseteq \mathrm{Aver}(\mathbf{P}, *)^{\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k,*)} = \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_{k+1}, *).$$

Hence, $\mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_{k+1}, *)^{(D,\mu)} \subseteq \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_{k+1}, *)$, which implies that (D,μ) belongs to $\mathrm{Aver}(\mathbf{L\Delta}^{\mathrm{p}}_{k+1}, \mathcal{F})$.

Similarly, we can prove the other inclusion. □

The assumption $\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *) = \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k+1}, *)$ is sufficient for the class $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$ to collapse to the kth level of the average low hierarchy.

Proposition 6.6.4 *Let $k > 0$ and $\mathcal{F}$ be a set of distributions. If $\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *) = \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k+1}, *)$, then $\mathrm{Aver}(\mathbf{L\Sigma}^{\mathrm{p}}_k, \mathcal{F}) = \mathrm{Aver}(\mathbf{NP}, \mathcal{F})$.*

Proof. Let us assume that $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *) = \mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, *)$. Let (D, μ) be an arbitrary distributional problem in $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$. We shall show that (D, μ) belongs to $\mathrm{Aver}(\mathbf{L}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$. Since $(D, \mu) \in \mathrm{Aver}(\mathbf{NP}, \mathcal{F})$, we have the following inclusions:

$$\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *) \subseteq \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)^{(D,\mu)} \subseteq \mathrm{Aver}(\mathbf{\Sigma}_{k+1}^{\mathrm{p}}, *).$$

By our assumption, it follows that $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)^{(D,\mu)} \subseteq \mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, *)$. This shows that (D, μ) is in $\mathrm{Aver}(\mathbf{L}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$. □

Unfortunately, we do not know any natural examples of distributional decision problems falling in the average low and high hierarchies. The search for such problems is a challenge.

Chapter 7

Quintessential Computability

7.1 Introduction

The most exciting aspect of this thesis is the attempt to investigate the notion of *quintessential computability*, first proposed by Schuler and Yamakami [97]. Throughout this thesis, we have developed the average-case complexity theory initiated by Levin. We know that average-case complexity theory is very sensitive to the choice of distributions. For example, if we take a distribution which decreases fast enough to 0, then all **NP** problems are polynomial time solvable on the average. Nevertheless, this type of extreme analysis does not capture the significant feature of average-case complexity theory.

Regarding Levin's question Dist($\mathbf{NP}, \mathbf{P}$-comp) $\subseteq$?Aver($\mathbf{P}, *$), Ben-David, Chor, Goldreich, and Luby [9] gave a partially negative answer by demonstrating that Dist($\mathbf{NP}, \mathbf{P}$-comp) $\not\subseteq$ Aver($\mathbf{P}, *$) unless $\mathbf{E} = \mathbf{NE}$. This result suggests one approach to open questions posed in average-case complexity theory: embedding average-case complexity classes into worst-case complexity theory in such a way that the embedding does not lose the complexity of these classes. The simplest solution is to focus on "rare instances" under "every" reasonable distribution. This notion was developed by Schuler and Yamakami [97] (suggested by Uwe Schöning).

In Section 7.2, we shall formally introduce the notion of "real $\mathcal{C}$ under $\mathcal{F}$." For a (worst-case) complexity class $\mathcal{C}$, "real $\mathcal{C}$ under $\mathcal{F}$" represents the class of sets which can be computable on average no matter what distributions are chosen from $\mathcal{F}$. The simple notation $\mathcal{C}_{\mathcal{F}}$ was invented by Schuler and Yamakami [97] to denote the class "real $\mathcal{C}$ under $\mathcal{F}$;" for example, $\mathbf{P}_{\mathbf{P}\text{-comp}}$ captures "real $\mathbf{P}$ under $\mathbf{P}$-comp." Using this notation, Levin's question Dist($\mathbf{NP}, \mathbf{P}$-comp) $\subseteq$?Aver($\mathbf{P}, *$) can be simply rephrased by the question $\mathbf{NP} \subseteq ?\mathbf{P}_{\mathbf{P}\text{-comp}}$ in the worst-case setting.

Based on this notion, we are able to introduce the *real polynomial-time hierarchy under* $\mathcal{F}$, $\{\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathcal{F}}, \mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}}, \mathbf{\Pi}_k^{\mathrm{p}}{}_{\mathcal{F}} \mid k > 0\}$, that captures the essentials of the average polynomial-time hierarchy under $\mathcal{F}$. By the definition, quintessentially computable classes have very different structures from their worst-case counterparts. For instance, it is unknown whether $\mathbf{P}_{\mathcal{F}}$ equals $\bigcup_{k>0} \mathrm{DTIME}(O(n^k))_{\mathcal{F}}$ in general, whereas $\mathbf{P}$ is

$\bigcup_{k>0} \mathrm{DTIME}(O(n^k))$. One of the exciting results here is that the real polynomial-time hierarchy under the set of recursive distributions indeed coincides with the worst-case polynomial-time hierarchy. In particular, $\mathbf{P}_{\mathbf{E}\text{-comp}}$ collapses to $\mathbf{P}$. These results support our main focus on feasible distributions in average-case analysis. Section 7.2 will formally define the notion of "real $\mathcal{C}$ under $\mathcal{F}$" and demonstrate the fundamental properties and the equivalence between the real polynomial-time hierarchy under **REC**-comp and the polynomial-time hierarchy.

Our interests are now in the real polynomial-time hierarchy under **P**-comp and its alternation counterpart, the *real polynomial-time alternation hierarchy under* **P**-comp, and we shall study its properties from the perspective of structural behaviors. The first question we want to ask ourselves is whether any level of the real polynomial-time hierarchy under **P**-comp contains sets which are hard to compute. Schuler [92] succeeded in constructing such a hard set within $\mathbf{P}_{\mathbf{P}\text{-comp}}$ which cannot be computed by a deterministic polynomial-time Turing machines. His method is further extended by Schuler and Yamakami [98] to the separation between $\mathrm{DTIME}(O(2^{c \cdot n}))$ and $\mathbf{P}_{\mathbf{P}\text{-comp}}$. In Section 7.3.1, we shall see how to construct hard sets within $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ and $\mathbf{\Sigma}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$, $k \in \mathbb{N}$, by using resource-bounded Kolmogorov complexity.

Immune sets and bi-immune sets are good examples of hard sets. The class **E**, for example, contains **P**-immune sets and **P**-bi-immune sets, and thus **E** is different from **P**. We shall see that $\mathbf{P}_{\mathbf{P}\text{-comp}}$ contains a **P**-immune set of an "arbitrary" density. Nonetheless, the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ or even its truth-table closure has no **P**-bi-immune sets, and as a consequence, the class turns out to be *small*, i.e., it has p-measure 0, from measure theoretic point of view. This consequence is intriguing in contrast with the fact that the Turing closure of $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is equal to the class **EXP**, which has p-measure 1.

In 1974, Book [15] first showed that **E** is *structurally* different from **NP**. He actually proved that **NP** enjoys the closure property under p-m-reductions, but **E** does not; therefore, **NP** cannot equal **E**. A similar structural property characterizes the classes in the real polynomial-time hierarchy under **P**-comp. We shall see in Section 7.5 that neither $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ nor $\mathbf{\Sigma}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ is closed under p-m-reductions, and consequently both $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ and $\mathbf{\Sigma}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ are *structurally* different from all worst-case complexity classes which are closed under p-m-reductions.

In Section 7.6, we shall look at the probabilistic classes $\mathbf{BPP}_{\mathbf{P}\text{-comp}}$ and $\mathbf{BPP}_{\mathbf{P}\text{-samp}}$, which are another example of well-studied quintessential complexity classes. Due to Impagliazzo and Levin [44] and Schuler and Watanabe [96], the question $\mathbf{NP} \subseteq ?\mathbf{BPP}_{\mathbf{P}\text{-comp}}$ is known to be equivalent to the question $\mathbf{NP} \subseteq ?\mathbf{BPP}_{\mathbf{P}\text{-samp}}$. This is not known for $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and $\mathbf{P}_{\mathbf{P}\text{-samp}}$.

The notion of random oracles was introduced by Bennett and Gill [8] in 1981 to prove that the probability of the event of a relativized **NP** coinciding with a relativized **P** is 0 when oracles are chosen at random. In Section 7.7, we shall show that neither of the inclusions, $\mathbf{NP} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$ nor $\mathbf{P}_{\mathbf{P}\text{-comp}} \subseteq \mathbf{NP}$, is possible relative to a random oracle.

Major Contributions. The author formalizes the notion of substantial computation on the average and shows the hardness and the structural properties of the classes in the real polynomial-time hierarchy under **P**-comp.

Propositions 7.2.5 and 7.2.10 show the inclusions among quintessential computable classes: e.g., $\mathbf{\Delta}_k^{\mathrm{p}} \subseteq \mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathcal{F}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathcal{F}} \subseteq \mathbf{\Delta}_k^{\mathrm{e}}$. A similar inclusion holds for the $\mathbf{\Sigma}_k^{\mathrm{p}}$ class.

Proposition 7.2.14 shows that, for any class $\mathcal{C}$ of the polynomial-time hierarchy, every set in $\mathcal{C}_{\mathcal{F}}$ is nearly-$\mathcal{C}$. As a direct consequence, Corollary 7.2.15 shows that if strong one-way functions exist, then **NP** is not included in $\mathbf{BPP}_{\mathcal{F}}$ for any set of distributions $\mathcal{F}$ which contains at least one supportive distribution.

Lemma 7.2.16 shows that if $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ has the sparse interpolation property, then $\mathrm{TALLY} \cap \mathcal{C}_{\mathcal{F}} \subseteq \mathcal{C}$. This lemma leads to several interesting consequences including, as in Corollary 7.2.19, $\mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathcal{F}}$ unless $\mathbf{\Delta}_k^{\mathrm{e}} = \mathbf{\Sigma}_k^{\mathrm{e}}$.

Theorem 7.2.23 shows that $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{REC}\text{-comp}} = \mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{REC}\text{-comp}} = \mathbf{\Delta}_k^{\mathrm{p}}$ for any k. Similarly for $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{REC}\text{-comp}}$ and $\mathbf{BPP}_{\mathbf{REC}\text{-comp}}$. In particular, as in Corollary 7.2.24, the class $\mathbf{P}_{\mathbf{E}\text{-comp}}$ equals **P**.

Theorem 7.3.2 shows that, for each constant $c > 0$, there exists a sparse set in $\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ but not in $\mathrm{ATIME}^{\Delta}(k, O(2^{cn}))$. A similar separation result holds for $\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$.

Proposition 7.3.4 shows that, assuming $\mathbf{P}_{\mathbf{P}\text{-samp}} \neq \mathbf{P}$, either $\mathbf{FP}^{\mathbf{E}} \not\subseteq \#\mathbf{P}$ or $\mathbf{NP} \not\subseteq \mathbf{BPP}$ holds.

Proposition 7.3.6 shows that for each constant $c > 0$, $\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is not included in $\mathrm{ATIME}^{\Delta}(k, O(2^{c \cdot n}))$-close. This immediately implies that $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{P}$-close.

Theorem 7.3.9 shows that, for each constant $c > 0$, $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is not included in $\mathbf{\Delta}_k^{\mathrm{p}}/cn$.

Theorem 7.4.1 shows that there exists a **P**-immune set in $\mathbf{P}_{\mathbf{P}\text{-comp}}$. The final claim of Proposition 7.4.5, that there is no **P**-bi-immune set in $\mathbf{P}_{\mathbf{P}\text{-comp}}$, comes from Schuler and Yamakami [98].

Proposition 7.5.10 shows that $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-samp}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ are closed under hp-m-reductions.

Lemma 7.5.9 shows that there exists an incomparable pair of sets A and B in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ with respect to hp-m-reducibility, namely, $A \not\leq_m^{\mathrm{hp}} B$ and $B \not\leq_m^{\mathrm{hp}} A$.

Proposition 7.5.8 shows that if every **P**-samplable distribution is avp-dominated by some **P**-computable distribution, then $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is closed downward under hp-m-reductions.

As for polynomially-bounded operators, Theorem 7.5.13 shows that the class $\exists^{\mathrm{p}} \cdot \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is not included in $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$. As its corollary (Corollary 7.5.14), $\exists^{\mathrm{p}} \cdot \mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{NP}_{\mathbf{P}\text{-comp}}$ if $\mathbf{P} = \mathbf{NP}$. Theorem 7.5.16 asserts a similar result about the probabilistic operator P^{p} that the class $\mathrm{P}^{\mathrm{p}} \cdot \mathbf{P}_{\mathbf{P}\text{-comp}}$ is not included in $\mathbf{P}_{\mathbf{P}\text{-comp}}$.

There is a series of random oracle separation results taken from Schuler and Yamakami [97, 98]. Proposition 7.7.4 shows that, relative to a random oracle, $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is different from $\mathbf{NP}_{\mathbf{P}\text{-comp}}$. In Theorem 7.7.6, it is shown that, relative to a random oracle, **NP** is not included in $\mathbf{P}_{\mathbf{P}\text{-comp}}$, whereas Theorem 7.7.7 shows that, relative to a random oracle, $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is not included in **PSPACE**.

7.2 Real Polynomial-Time Hierarchy

Average-case complexity theory has given us a different perspective from worst-case complexity theory about what is hard to compute. Some **NP**-complete problems have been already categorized as relatively "easy" on the average. This is one of the reasons that Levin asked whether *all* **NP**-complete problems are "easy" on the average.

As shown by Ben-David, Chor, Goldreich, and Luby [9], Levin's question is closely related to the $\mathbf{E} =?\mathbf{NE}$ question in worst-case complexity theory. Is there any general way we can discuss other open questions in average-case complexity theory in terms of worst-case complexity classes ? Or more bravely, can we embed our average-case complexity classes into the world of worst-case complexity classes without losing any significant feature of their average-case complexity ? One answer was presented by Schuler and Yamakami [97] in 1992. They introduced a new notion, called *real* $\mathcal{C}$ *under* $\mathcal{F}$, that enables us to treat an average-case complexity class as a worst-case complexity class.

This section will begin with the notion *real* $\mathcal{C}$ *under* $\mathcal{F}$ and then introduce the *real polynomial-time hierarchy*.

7.2.1 The Notion of "Real $\mathcal{C}$ under $\mathcal{F}$"

At a conference in 1992, Schuler and Yamakami [97] proposed a way that we can bring average-case complexity classes back to the worst-case complexity world. The idea is that rather than argue the average behavior of an algorithm with respect to each individual distribution, we wish to extract the hardest instances under *every* distribution. Let us consider a set A in $\mathbf{P}$. The set A is not only computable in polynomial time, but also computable in polynomial time on μ-average under every distribution μ. In other words, A is "easy" to compute regardless of the probability with which each instance occurs. What kind of instances are "easy" to compute on average under *all* reasonable distributions like $\mathbf{P}$-computable distributions ? We shall formalize the collection of such instances in a more general way.

Formally, we introduce the general notion of "real $\mathcal{C}$ under $\mathcal{F}$."

Definition 7.2.1 (Real $\mathcal{C}$ under $\mathcal{F}$) [97] Let $\mathcal{C}$ be a complexity class and let $\mathcal{F}$ be a class of distributions. Assume that $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ is defined. The class *real* $\mathcal{C}$ *under* $\mathcal{F}$, symbolically $\mathcal{C}_{\mathcal{F}}$, is the class of sets D such that $(D, \mu) \in \mathrm{Aver}(\mathcal{C}, *)$ for every $\mu \in \mathcal{F}$.

This new notion formalizes a significant feature of the associated average-case complexity classes. The next proposition indicates its importance.

***Proposition 7.2.2* [97]** *Let* $\mathrm{Dist}(\mathcal{C}, \mathcal{F})$ *and* $\mathrm{Aver}(\mathcal{D}, \mathcal{F})$ *be any randomized and average-case complexity classes, respectively. Then,* $\mathcal{C} \subseteq \mathcal{D}_{\mathcal{F}}$ *if and only if* $\mathrm{Dist}(\mathcal{C}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathcal{D}, \mathcal{F})$.

Proof. Assume that $\mathcal{C} \subseteq \mathcal{D}_{\mathcal{F}}$ and (A, μ) is in $\mathrm{Dist}(\mathcal{C}, \mathcal{F})$. From the fact that A belongs to $\mathcal{D}_{\mathcal{F}}$, it follows that $(A, \mu) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$. Conversely, assume that $\mathrm{Dist}(\mathcal{C}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathcal{D}, *)$. Let D be any a set in $\mathcal{C}$. For every $\mu \in \mathcal{F}$, since $(D, \mu) \in \mathrm{Dist}(\mathcal{C}, \mathcal{F})$, we obtain $(D, \mu) \in \mathrm{Aver}(\mathcal{D}, \mathcal{F})$. Hence, D belongs to $\mathcal{D}_{\mathcal{F}}$. □

For most classes $\mathcal{C}$, we immediately conclude the inclusion $\mathcal{C} \subseteq \mathcal{C}_{\mathcal{F}}$, since $\mathrm{Dist}(\mathcal{C}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathcal{C}, \mathcal{F})$.

An advantage of Proposition 7.2.2 is that Levin's question $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \subseteq ?\mathrm{Aver}(\mathbf{P}, *)$ can be simply rephrased as follows: "Is $\mathbf{NP}$ included in $\mathbf{P}_{\mathbf{P}\text{-comp}}$?" In an attempt to answer his question, we

must make a careful study of the quintessential complexity class $\mathbf{P}_{\mathbf{P}\text{-comp}}$. More generally, the question $\mathrm{Dist}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathbf{P}\text{-comp}) \subseteq ?\mathrm{Aver}(\mathbf{\Delta}_k^{\mathrm{p}}, *)$ is translated into the question $\mathbf{\Sigma}_k^{\mathrm{p}} \subseteq ?\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ in the worst-case setting.

From Definition 7.2.1, we obtain quintessential complexity classes $\mathbf{NP}_{\mathcal{F}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}}$, where $k \in \mathbb{N}$. Recall the definition of the average-case complexity classes, $\mathrm{Aver}(\text{co-}\mathbf{NP}, \mathcal{F})$ and $\mathrm{Aver}(\mathbf{\Pi}_k^{\mathrm{p}}, \mathcal{F})$, $k \in \mathbb{N}$. These classes are conventionally defined to be the complements of the original defined-by-machine classes $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$ and $\mathrm{Aver}(\mathbf{\Sigma}_k^{\mathrm{p}}, \mathcal{F})$. Adapting Definition 7.2.1, we are able to "define" the classes $(\text{co-}\mathbf{NP})_{\mathcal{F}}$ and $\mathbf{\Pi}_k^{\mathrm{p}}{}_{\mathcal{F}}$. Nevertheless, do these classes conflict with the definition of the complement class ? More specifically, do they equal the complements of $\mathbf{NP}_{\mathcal{F}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}}$? The following lemma shows that Definition 7.2.1 does not conflict with the complement classes $\mathrm{Aver}(\text{co-}\mathcal{C}, \mathcal{F})$ in general.

Lemma 7.2.3 *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes and $\mathcal{F}$ be a set of distributions.*

1. *If $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ is defined, then $\text{co-}\mathcal{C}_{\mathcal{F}} = (\text{co-}\mathcal{C})_{\mathcal{F}}$.*
2. *If $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ and $\mathrm{Aver}(\mathcal{D}, \mathcal{F})$ are defined, then $\mathcal{C}_{\mathcal{F}} \cap \mathcal{D}_{\mathcal{F}} = (\mathcal{C} \cap \mathcal{D})_{\mathcal{F}}$.*

Proof. (1) For any set $A \in \text{co-}\mathcal{C}_{\mathcal{F}}$, we have $\overline{A} \in \mathcal{C}_{\mathcal{F}}$. For every distribution $\mu \in \mathcal{F}$, $(\overline{A}, \mu) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$. By our assumption, this is equivalent to the statement that $(A, \mu) \in \mathrm{Aver}(\text{co-}\mathcal{C}, \mathcal{F})$ for every $\mu \in \mathcal{F}$. Thus, we have $A \in (\text{co-}\mathcal{C})_{\mathcal{F}}$.

Conversely, assume that $A \in (\text{co-}\mathcal{C})_{\mathcal{F}}$. We have $(A, \mu) \in \mathrm{Aver}(\text{co-}\mathcal{C}, \mathcal{F})$ for all distributions μ in $\mathcal{F}$, and thus $(\overline{A}, \mu) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$. Hence, $\overline{A} \in \mathcal{C}_{\mathcal{F}}$. In other words, $A \in \text{co-}\mathcal{C}_{\mathcal{F}}$.

(2) By a similar, simple argument. □

In particular, we have the equalities $\mathbf{\Pi}_k^{\mathrm{p}}{}_{\mathcal{F}} = \text{co-}\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}} \cap \mathbf{\Pi}_k^{\mathrm{p}}{}_{\mathcal{F}} = (\mathbf{\Sigma}_k^{\mathrm{p}} \cap \mathbf{\Pi}_k^{\mathrm{p}})_{\mathcal{F}}$ for all $k \geq 0$ and for any set $\mathcal{F}$ of distributions.

We make the remark here that although we have seen in Corollary 3.5.18 that $\mathrm{Aver}(\mathbf{P}, \mathbf{P}\text{-comp}) \not\subseteq \mathrm{Dist}(\mathbf{NP}, *)$, we do not know whether $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{NP}$.

The following inclusion follows immediately from Corollary 3.4.7.

Lemma 7.2.4 *Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be two sets of distributions and let $\mathcal{C}$ be any class in the polynomial-time hierarchy. If every distribution in $\mathcal{F}_1$ is avp-dominated by some distribution in $\mathcal{F}_2$, then $\mathcal{C}_{\mathcal{F}_2} \subseteq \mathcal{C}_{\mathcal{F}_1}$.*

We remark here that $\mathbf{PH}_{\mathcal{F}}$ is *not* defined as the union of all $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}}$ for any $k \in \mathbb{N}$. Hence, despite Proposition 6.4.6, we cannot simply conclude that $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}} = \mathbf{\Sigma}_{k+1}^{\mathrm{p}}{}_{\mathcal{F}}$ implies $\mathbf{PH}_{\mathcal{F}} = \mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathcal{F}}$.

We next locate the newly defined classes $\mathbf{P}_{\mathcal{F}}$, $\mathbf{NP}_{\mathcal{F}}$, $\mathbf{BPP}_{\mathcal{F}}$, and $\mathbf{PSPACE}_{\mathcal{F}}$ in the worst-case world.

Proposition 7.2.5 **[97]** *Let $\mathcal{F}$ be any set of distributions which contains the standard distribution.*

1. $\mathbf{P} \subseteq \mathbf{P}_{\mathcal{F}} \subseteq \mathbf{E}$.
2. $\mathbf{NP} \subseteq \mathbf{NP}_{\mathcal{F}} \subseteq \mathbf{NE}$.

3. $\mathbf{BPP} \subseteq \mathbf{BPP}_{\mathcal{F}} \subseteq \mathbf{BPE}$.

4. $\mathbf{PSPACE} \subseteq \mathbf{PSPACE}_{\mathcal{F}} \subseteq \mathbf{ESPACE}$.

Proof. We give only the proof of (1) since the rest of the claims follow by a similar argument. Since $\mathrm{Dist}(\mathbf{P}, \mathcal{F}) \subseteq \mathrm{Aver}(\mathbf{P}, \mathcal{F})$, we have $\mathbf{P} \subseteq \mathbf{P}_{\mathcal{F}}$. To show that $\mathbf{P}_{\mathcal{F}} \subseteq \mathbf{E}$, we let A be any set in $\mathbf{P}_{\mathcal{F}}$. Since $(A, \nu_{\mathrm{stand}}) \in \mathrm{Aver}(\mathbf{P}, \mathcal{F})$, there exist an increasing polynomial p and a deterministic Turing machine M which is p-time bounded on ν_{stand}-average such that M computes A. Recall that $\hat{\nu}_{\mathrm{stand}}(x) \geq \frac{1}{8(|x|+1)^2 \cdot 2^{|x|}}$. It clearly holds that, for almost all x,

$$\mathrm{Time}_M(x) \leq p(|x|/\hat{\nu}_{\mathrm{stand}}(x)) \leq p(8|x| \cdot (|x|+1)^2 \cdot 2^{|x|}) \leq 2^{c|x|}$$

for an appropriate constant $c > 0$. Therefore, A belongs to $\mathrm{DTIME}(O(2^{cn}))$, which is a subset of $\mathbf{E}$. □

As a final note in this introductory subsection, we wish to demonstrate that the class "real $\mathbf{P}$ under FLAT" is not a large class, where FLAT denotes the collection of all flat distributions. We note that whether $\mathbf{NP} \subseteq \mathbf{P}_{\mathrm{FLAT}}$ is an open question, which is related to the incompleteness of distributional problems with flat distributions. We now recall that $\mathbf{SUBEXP} = \bigcap_{\epsilon > 0} \mathrm{DTIME}(O(2^{n^\epsilon}))$.

Lemma 7.2.6 $\mathbf{P}_{\mathrm{FLAT}} \subseteq \mathbf{SUBEXP}$.

Proof. Assume that A is in $\mathbf{P}_{\mathrm{FLAT}}$. Let μ be a distribution such that $\hat{\mu}(x) \leq 2^{-|x|^{\epsilon(|x|)}}$ for almost all x, where $\epsilon(n) = 1/\lfloor \log n \rfloor$. This distribution is flat because ϵ is decreasing. By the choice of A, $(A, \mu) \in \mathrm{Aver}(\mathbf{P}, *)$. There is a deterministic Turing machine M computing A in time p on μ-average, where p is a function from Σ^* to $\mathbb{R}^+$. Choose constants $c, k > 0$ such that $p(z) \leq c \cdot z^k$ for almost all z.

Let m be any positive integer. Then, for any sufficiently large x,

$$\begin{aligned} \mathrm{Time}_M(x) &\leq p(|x|/\hat{\mu}(x)) \leq p(|x| \cdot 2^{|x|^{\epsilon(|x|)}}) \\ &\leq c \cdot (|x| \cdot 2^{|x|^{\epsilon(|x|)}}) \leq 2^{(k+1)|x|^{\epsilon(|x|)}} \\ &\leq 2^{|x|^{1/m}}. \end{aligned}$$

Thus, $A \in \mathrm{DTIME}(O(2^{n^{1/m}}))$. Since m is arbitrary, $A \in \mathbf{SUBEXP}$ follows. □

The converse is unlikely to hold.

7.2.2 Real Polynomial-Time Hierarchy

The notion of quintessential computability enables us to translate all average-case complexity classes into worst-case complexity classes. In particular, we can naturally translate the average polynomial-time hierarchy into its quintessential counterpart. We call such a hierarchy the *real polynomial-time hierarchy*. This subsection will study its structural properties.

The formal definition of the real polynomial-time hierarchy is given below.

Definition 7.2.7 (Real Polynomial-Time Hierarchy under $\mathcal{F}$) [97] Let $\mathcal{F}$ be any set of distributions. The *real polynomial-time hierarchy under* $\mathcal{F}$ consists of $\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}$, $\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}$, and $\mathbf{\Pi}^{\mathrm{p}}_{k}\mathcal{F}$ for all natural numbers k. Let $\mathbf{PH}_{\mathcal{F}}$ be the collection of all sets A such that (A,μ) is in $\mathrm{Aver}(\mathbf{PH},\mathcal{F})$ for all distributions $\mu\in\mathcal{F}$.

Immediately from the facts that $\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k},\mathcal{F})\subseteq\mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_{k+1},\mathcal{F})\subseteq\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k+1},\mathcal{F})$ and $\mathbf{\Pi}^{\mathrm{p}}_{k}\mathcal{F}=\text{co-}\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}$, it follows that:

Lemma 7.2.8 *For all $k\geq 0$, $\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}\cup\mathbf{\Pi}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Delta}^{\mathrm{p}}_{k+1}\mathcal{F}\subseteq\mathbf{\Sigma}^{\mathrm{p}}_{k+1}\mathcal{F}\cap\mathbf{\Pi}^{\mathrm{p}}_{k+1}\mathcal{F}$.*

Analogous to the real polynomial-time hierarchy, we can define the *real polynomial-time alternation hierarchy* using the average polynomial-time alternation hierarchy.

Definition 7.2.9 (Real Polynomial-Time Alternation Hierarchy under $\mathcal{F}$) Let $\mathcal{F}$ be any set of distributions. The *real polynomial-time alternation hierarchy under* $\mathcal{F}$ consists of $\mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}$, $\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}$, and $\mathbf{A}\mathbf{\Pi}^{\mathrm{p}}_{k}\mathcal{F}$ for all $k\in\mathbb{N}$. Let $\mathbf{APH}_{\mathcal{F}}$ be the collection of all sets A such that (A,μ) is in $\mathrm{Aver}(\mathbf{APH},\mathcal{F})$ for all distributions $\mu\in\mathcal{F}$.

Because of domination conditions imposed on oracle Turing machines, the real polynomial-time hierarchy forms a sub-hierarchy of the linear exponential-time alternation hierarchy, $\{\mathbf{\Delta}^{\mathrm{e}}_{k},\mathbf{\Sigma}^{\mathrm{e}}_{k},\mathbf{\Pi}^{\mathrm{e}}_{k}\mid k>0\}$.

Proposition 7.2.10 *Let $k>0$ and let $\mathcal{F}$ be any set of distributions which contains the standard distribution ν_{stand}.*

1. $\mathbf{\Delta}^{\mathrm{p}}_{k}\subseteq\mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Delta}^{\mathrm{e}}_{k}$.
2. $\mathbf{\Sigma}^{\mathrm{p}}_{k}\subseteq\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Sigma}^{\mathrm{e}}_{k}$.

Proof. Recall that $\mathrm{Dist}(\mathbf{\Delta}^{\mathrm{p}}_{k},\mathcal{F})\subseteq\mathrm{Aver}(\mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_{k},\mathcal{F})$ and $\mathrm{Dist}(\mathbf{\Sigma}^{\mathrm{p}}_{k},\mathcal{F})\subseteq\mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{k},\mathcal{F})$ for all $k>0$. Hence, $\mathbf{\Delta}^{\mathrm{p}}_{k}\subseteq\mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}$ and $\mathbf{\Sigma}^{\mathrm{p}}_{k}\subseteq\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}$. The inclusions $\mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}$ and $\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}$ come from Corollary 6.5.3. In the following, we show the rest of the claim, namely, $\mathbf{\Delta}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Delta}^{\mathrm{e}}_{k}$ and $\mathbf{\Sigma}^{\mathrm{p}}_{k}\mathcal{F}\subseteq\mathbf{\Sigma}^{\mathrm{e}}_{k}$.

It suffices to show that $(A,\nu_{\mathrm{stand}})\in\mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_{k},*)$ implies $A\in\mathbf{\Sigma}^{\mathrm{e}}_{k}$. This proceeds by induction on k. The base case $k=1$ follows from Lemma 7.2.5(2). Let $k\geq 2$. Let (A_i,μ_i), $1\leq i\leq k$, be a distributional problem, where $A_1=A$ and $\mu_1=\nu_{\mathrm{stand}}$. Assume that $(A_i,\mu_i)\in\mathrm{Aver}(\mathbf{NP},*)^{(A_{i+1},\mu_{i+1})}$ via an oracle machine M_i for all i with $1\leq i<k$. Assume also that A_k is recognized by a Turing machine N which runs in polynomial time on μ_k-average. Let $g_k(x)=\mathrm{Time}_N(x)$, $g_i(x)=\min_{y\in\mathrm{Flip}(M_i,A_{i+1},x)}\sum_{z\in Q(M_i,A_{i+1},x,y)}g_{i+1}(z)$. It is not difficult by induction on i to show that g_1 is polynomial on μ_1-average, since g_k is polynomial on μ_k-average.

First we construct alternating Turing machines M'_i, $1\leq i\leq k$, as in the proof that $\mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{i}=\mathbf{\Sigma}^{\mathrm{p}}_{i}$. More precisely, the machine M'_i is defined as follows: on input x, M'_i simulates in an existential state M_i on the same input except for oracle queries; if M_i queries z, then M'_i guesses its oracle answer $ans(z)$ and continue the simulation; if M_i reaches an accepting configuration, it sequentially simulates M'_{i+1} on all z with $ans(z)=0$;

then in a universal state, M_i' simulates M_{i+1}' on all z with $ans(z) = 0$; M_i' enters an accepting configuration exactly when M_{i+1}' reaches a rejecting configuration. Otherwise, M_i' enters a rejecting configuration. It is obvious that each M_i' is an alternating Turing machine with $(k - i + 1)$-alternation. Moreover, assuming $x \in L(M_i')$, the length of the minimal subtree of the computation tree of M_i' on x which contains only "yes"-configurations is bounded by $c \cdot g_i(x)$, where c is a positive constant not depending on the choice of x.

Second we define a new machine M as follows: on input x, M simulates M_1' on x in time $2^{c|x|+c}$; on each computation path, if M_1' does not reach any halting configuration, then M enters a rejecting configuration. Since g_1 is polynomial on $\nu_{\rm stand}$-average, $x \in A$ if and only if $x \in L(M)$. Hence, we have $A \in \mathbf{\Sigma}_k^{\rm e}$.

The case for $\mathbf{\Delta}_k^{\rm p}{}_{\mathcal{F}}$ is shown analogously. □

In worse-case complexity theory, **PH** is the union of all sets in $\mathbf{\Sigma}_k^{\rm p}$ for any $k > 0$. Nevertheless, we have no proof that $\mathbf{PH}_{\mathcal{F}} = \bigcup_{k>0} \mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}}$. On the other hand, can we show that $\mathbf{PH}_{\mathcal{F}}$ differs from $\bigcup_{k>0} \mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}}$? Since it is still possible that $\mathbf{P}_{\mathcal{F}} = \mathbf{PH}_{\mathcal{F}}$, the separation between $\mathbf{PH}_{\mathcal{F}}$ and $\bigcup_{k>0} \mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}}$ seems difficult to prove. This situation is similar to the question of $\mathrm{ATIME}(n^{O(1)})$ versus $\bigcup_{k>0} \mathrm{ATIME}^{\Sigma}(k, n^{O(1)})$. Because $\bigcup_{k>0} \mathrm{ATIME}^{\Sigma}(k, n^{O(1)})$ coincides with **PH** while $\mathrm{ATIME}(n^{O(1)})$ equals **PSPACE**, we do not know whether $\mathbf{PH} = \mathbf{PSPACE}$. Therefore, as mentioned before, we cannot conclude that if $\mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}} = \mathbf{\Sigma}_{k+1}^{\rm p}{}_{\mathcal{F}}$, then $\mathbf{PH}_{\mathcal{F}} = \mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}}$.

We have seen that $\mathbf{\Delta}_k^{\rm p} = \mathbf{\Sigma}_k^{\rm p}$ implies $\mathrm{Aver}(\mathbf{A}\mathbf{\Delta}_k^{\rm p}, \mathcal{F}) = \mathrm{Aver}(\mathbf{A}\mathbf{\Sigma}_k^{\rm p}, \mathcal{F})$. Hence:

Lemma 7.2.11 *Let $k > 0$ and let $\mathcal{F}$ be a set of distributions. If $\mathbf{\Delta}_k^{\rm p} = \mathbf{\Sigma}_k^{\rm p}$, then $\mathbf{A}\mathbf{\Delta}_k^{\rm p}{}_{\mathcal{F}} = \mathbf{A}\mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}}$.*

Last, we shall demonstrate some basic closure properties, under set operators, of classes in the real polynomial-time hierarchy under **P**-comp.

Lemma 7.2.12 *Let $\mathcal{C} \in \{\mathbf{\Delta}_k^{\rm p}, \mathbf{\Sigma}_k^{\rm p}, \mathbf{\Pi}_k^{\rm p} \mid k > 0\}$. If A and B in $\mathcal{C}_{\mathbf{P}\text{-comp}}$, then $A \cap B$, $A \cup B$, $A - B$, and $A \oplus B$ are in $\mathcal{C}_{\mathbf{P}\text{-comp}}$.*

Proof. Let $k > 0$. Here we shall show the closure property of $\mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}}$ under $\oplus$. Let us assume that A and B are in $\mathbf{\Sigma}_k^{\rm p}{}_{\mathcal{F}}$: that is, $(A, \mu) \in \mathrm{Aver}(\mathbf{\Sigma}_k^{\rm p}, \mathbf{P}\text{-comp})$ and $(B, \mu) \in \mathrm{Aver}(\mathbf{\Sigma}_k^{\rm p}, \mathbf{P}\text{-comp})$ for all distributions μ in **P**-comp. Take an arbitrary distribution $\mu \in \mathbf{P}$-comp. We assume that $\hat{\mu}(\lambda) = 0$ for simplicity. Let c_0 and c_1 be defined as

$$c_b = \begin{cases} 2 & \text{if } \sum_x \hat{\mu}(bx) > \frac{1}{2}, \\ 1 & \text{otherwise,} \end{cases}$$

where $b \in \{0, 1\}$. We next define μ' as $\hat{\mu}'(\lambda) = 0$, and for each nonempty z,

$$\hat{\mu}'(z) = \begin{cases} \frac{1}{c_0} \cdot \hat{\mu}(bx) & \text{if } z = bx \text{ for some } b \in \{0,1\} \text{ and } x \in \Sigma^+, \\ \frac{1}{2} - \sum_{w : w \neq \lambda} \hat{\mu}_0(bw) & \text{if } z = b \in \{0,1\}. \end{cases}$$

The function μ' becomes a distribution and satisfies $\hat{\mu}'(0\Sigma^*) = \hat{\mu}'(1\Sigma^*) = \frac{1}{2}$. It is also easy to see that $\mu \preceq^{\rm p} \mu'$ because $\hat{\mu}(x) \leq 2 \cdot \hat{\mu}'(x)$. Hence, $(A \oplus B, \mu) \leq_m^{\rm p} (A \oplus B, \mu')$.

Define $\hat{\mu}_A(x) = 2 \cdot \hat{\mu}'(0x)$ and $\hat{\mu}_B(x) = 2 \cdot \hat{\mu}'(1x)$ for all x. Notice that $\mu' = \mu_A \oplus \mu_B$. By the definition of μ', both μ_A and μ_B are $\mathbf{P}$-computable. Thus, we conclude that (A, μ_A) and (B, μ_B) are in $\mathrm{Aver}(\mathcal{C}, \mathbf{P}\text{-comp})$. Lemma 6.4.7 yields the conclusion that $(A \oplus B, \mu_A \oplus \mu_B) \in \mathrm{Aver}(\mathcal{C}, \mathbf{P}\text{-comp})$, which means $(A \oplus B, \mu') \in \mathrm{Aver}(\mathcal{C}, \mathbf{P}\text{-comp})$. Since $\mathrm{Aver}(\mathcal{C}, \mathbf{P}\text{-comp})$ is closed under $\leq^{\mathrm{p}}_m$-reductions, then $(A \oplus B, \mu)$ is in $\mathrm{Aver}(\mathcal{C}, \mathbf{P}\text{-comp})$.

Since μ is arbitrary, we get $A \oplus B \in \mathcal{C}_{\mathbf{P}\text{-comp}}$. □

7.2.3 Nearly-$\mathbf{\Sigma}^{\mathrm{p}}_k$ and Nearly-$\mathbf{\Delta}^{\mathrm{p}}_k$ Sets

In Section 4.7, we introduced the notions of "nearly-**RP**" and "nearly-**BPP**" sets. In a similar fashion, we can extend this notion and introduce the new notions of "nearly-$\mathbf{\Sigma}^{\mathrm{p}}_k$" and "nearly-$\mathbf{\Delta}^{\mathrm{p}}_k$" sets. Our goal here is to prove that every set in $\mathbf{\Sigma}^{\mathrm{p}}_{k\mathcal{F}}$ ($\mathbf{\Delta}^{\mathrm{p}}_{k\mathcal{F}}$, resp.) is nearly-$\mathbf{\Sigma}^{\mathrm{p}}_k$ (nearly-$\mathbf{\Delta}^{\mathrm{p}}_k$, resp.).

Definition 7.2.13 (Nearly-$\mathbf{\Sigma}^{\mathrm{p}}_k$ and Nearly-$\mathbf{\Delta}^{\mathrm{p}}_k$ Sets) A set A is *nearly-$\mathbf{\Sigma}^{\mathrm{p}}_k$* if, for every polynomial p, there exist a set S and a polynomial-time alternating Turing machine M whose alternation is at most k starting with an existential state such that (i) $x \in A - S$ implies $M(x) = 1$; (ii) $x \in \overline{A} - S$ implies $M(x) = 0$; and (iii) $\mathbf{Pr}_n[x \in S] < \frac{1}{p(n)}$ for almost all n. Similarly, a notion of *"nearly-$\mathbf{\Delta}^{\mathrm{p}}_k$"* is defined by using an alternating Turing machine with a semi-deterministic process.

Proposition 7.2.14 *Let $\mathcal{F}$ be a set of distributions such that $\nu_{\mathrm{stand}} \in \mathcal{F}$. Let $\mathcal{C}$ be one of the following classes, $\mathbf{\Delta}^{\mathrm{p}}_k$, $\mathbf{\Sigma}^{\mathrm{p}}_k$, $k \in \mathbb{N}$, **BPP**, and **RP**. Then, every set in $\mathcal{C}_{\mathcal{F}}$ is nearly-$\mathcal{C}$.*

Proof. First we shall show the case $\mathcal{C} = \mathbf{\Sigma}^{\mathrm{p}}_k$. The other case $\mathcal{C} = \mathbf{\Delta}^{\mathrm{p}}_k$ follows similarly. Let A be an arbitrary set in $\mathbf{\Sigma}^{\mathrm{p}}_{k\mathcal{F}}$. Note that $(A, \nu_{\mathrm{stand}}) \in \mathrm{Aver}(\mathbf{\Sigma}^{\mathrm{p}}_k, *)$. By Proposition 6.4.12, we have two sets $C_0 \in \mathbf{\Sigma}^{\mathrm{p}}_k$ and $C_1 \in \mathbf{\Pi}^{\mathrm{p}}_k$ such that $C_0 \subseteq A$, $C_1 \subseteq \overline{A}$, and $\|C^n_0 \cup C^n_1\| \geq (1 - 1/p(n)) \cdot 2^n$ for all $n \in \mathbb{N}$. Let $S = \Sigma^* - (C_0 \cup C_1)$. We then have $\frac{\|S^n\|}{2^n} < \frac{1}{p(n)}$ for almost all n, and also we have $C_0 = A - S$ and $C_1 = \overline{A} - S$. Therefore, A is in nearly-$\mathbf{\Sigma}^{\mathrm{p}}_k$.

Next we shall prove the proposition for $\mathcal{C} = \mathbf{BPP}$. Assume that $A \in \mathbf{BPP}_{\mathcal{F}}$. Since $\nu_{\mathrm{stand}} \in \mathcal{F}$, we have $(A, \nu_{\mathrm{stand}}) \in \mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$. For convenience, write ν for ν_{stand}. By the definition of $\mathrm{Aver}(\mathbf{BPP}, \mathcal{F})$, there is a bounded-error probabilistic Turing machine M computing A in time p on ν-average, where p is an increasing polynomial. In particular, for the random-input domain Γ_M associated to M, $\hat{\nu}_{\Gamma_M}(\{(x, s) \mid \mathrm{Time}_M(x; s) > p(|x| \cdot r)\}) < 1/r$ for any real number $r > 0$. By Lemma 3.3.21, it follows that $\hat{\nu}(\{x \mid \mathrm{Time}^*_M(x) > p(|x| \cdot r)\}) < 1/r$. We can assume by the Amplification Lemma that $\mathbf{Pr}_M[M(x) = A(x)] \geq 5/6$ for all x.

Take any polynomial q, and let $q'(n) = p(n \cdot 40(n+1)^2 \cdot q(n))$. We then define

$$S = \left\{ x \;\middle|\; \mathbf{Pr}_s[\mathrm{Time}_M(x; s) > q'(|x|) \mid s \in \Gamma_M] \geq \frac{1}{5} \right\}.$$

Thus, if $x \in S$, then $5 \cdot \sum_{s \in \Gamma_M(x)} 2^{-|s|} \cdot [\mathrm{Time}_M(x; s) > q'(|x|)] \geq 1$.

First we shall show that the density of S is not so large. Fix $n \in \mathbb{N}$.

$$\begin{aligned}
\mathbf{Pr}_x[x \in S \mid x \in \Sigma^n] &\leq \sum_{x \in S \cap \Sigma^n} 2^{2\mathrm{llog}(n)+1} \cdot \hat{\nu}(x) \cdot 5 \cdot \mathbf{Pr}_s[\mathrm{Time}_M(x;s) > q'(n) \mid s \in \Gamma_M(x)] \\
&\leq 8(n+1)^2 \cdot 5 \cdot \sum_{x \in S \cap \Sigma^n} \hat{\nu}(x) \cdot \mathbf{Pr}_s[\mathrm{Time}_M(x;s) > q'(n) \mid s \in \Gamma_M(x)] \\
&= 40(n+1)^2 \cdot \hat{\nu}_{\Gamma_M}(\{(x,s) \mid x \in \Sigma^n \wedge \mathrm{Time}_M(x;s) > q'(n)\}) \\
&\leq 40(n+1)^2 \cdot \frac{1}{40(n+1)^2 \cdot q(n)} = \frac{1}{q(n)}.
\end{aligned}$$

Next we define a randomized Turing machine N as follows: on input x, simulate M on x in time $q'(|x|)$; if a computation does not terminate, then simply reject the input. Note that the error probability is less than $1/6$. We shall show that N correctly computes A on most inputs. For x not in S,

$$\begin{aligned}
\mathbf{Pr}_N[N(x) = A(x)] &= \mathbf{Pr}_s[\mathrm{Time}_M(x;s) \leq q'(|x|) \wedge M(x;s) = A(x) \mid s \in \Gamma_M(x)] \\
&\geq \mathbf{Pr}_s[\mathrm{Time}_M(x;s) \leq q'(|x|) \mid s \in \Gamma_M(x)] \cdot \mathbf{Pr}_s[M(x;s) = A(x) \mid s \in \Gamma_M(x)] \\
&\geq \left(1 - \frac{1}{5}\right) \cdot \frac{5}{6} = \frac{2}{3}.
\end{aligned}$$

Hence, A is nearly-$\mathbf{BPP}$. The case for $\mathcal{C} = \mathbf{RP}$ is similar. □

Corollary 7.2.15 *If strong one-way functions exist, then* $\mathbf{NP} \not\subseteq \mathbf{BPP}_{\mathcal{F}}$ *for any set* $\mathcal{F}$ *of distributions which includes* ν_{stand}.

Proof. Suppose that there is a strong one-way function. Assume also that $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathcal{F}}$ for some $\mathcal{F}$ with $\nu_{\mathrm{stand}} \in \mathcal{F}$. By Proposition 7.2.14, every $\mathbf{NP}$ set is nearly-$\mathbf{BPP}$. Proposition 2.6.4 shows that there is no strong one-way function. This is a contradiction; hence, $\mathbf{NP} \not\subseteq \mathbf{BPP}_{\mathcal{F}}$. □

7.2.4 Collapsing Classes

Let us return to Levin's original question of whether $\mathbf{NP} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$. As discussed in the previous section, we can now raise the more general question of whether $\mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_{k\,\mathcal{F}}^{\mathrm{p}}$ holds for some $\mathcal{F}$. Clearly if $\mathbf{P} = \mathbf{NP}$, then $\mathbf{NP}$ is included in $\mathbf{P}_{\mathcal{F}}$ for all $\mathcal{F}$. Ben-David *et al.* [9] first gave a partial answer to this question by showing that $\mathrm{Dist}(\mathbf{NP}, \mathbf{P}\text{-comp}) \not\subseteq \mathrm{Aver}(\mathbf{P}, *)$ if $\mathbf{E} \neq \mathbf{NE}$. In other words, $\mathbf{NP} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$ implies $\mathbf{E} = \mathbf{NE}$. Note that $\mathbf{E} = \mathbf{NE}$ if and only if $\mathrm{TALLY} \cap \mathbf{NP} \subseteq \mathbf{P}$ [14]. Hence, Ben-David et al. actually showed that $\mathrm{TALLY} \cap \mathbf{P}_{\mathbf{P}\text{-comp}} \subseteq \mathbf{P}$.

We shall generalize this result and show that the tally part of any real average complexity class collapses to its worst-case counterparts. First we state a technical lemma. Recall the standard distribution ν_{tally}. In the rest of this section, we use this distribution.

Lemma 7.2.16 **[97]** *Let* $\mathcal{C}$ *be a complexity class and let* $\mathcal{F}$ *be a set of distributions which contains* ν_{tally}. *Assume that* $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ *is defined. If* $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$ *has the sparse interpolation property, then* $\mathrm{TALLY} \cap \mathcal{C}_{\mathcal{F}} \subseteq \mathcal{C}$.

Proof. Let A be in TALLY $\cap\, \mathcal{C}_{\mathcal{F}}$. We note that $(A, \nu_{\mathrm{tally}}) \in \mathrm{Aver}(\mathcal{C}, \mathcal{F})$ since $\nu_{\mathrm{tally}} \in \mathcal{F}$. By the sparse interpolation property for $\mathrm{Aver}(\mathcal{C}, \mathcal{F})$, there exists a set B in $\mathcal{C}$ such that $A \cap \{0\}^* \subseteq B \subseteq A$. Since $A \subseteq \{0\}^*$, we have $A = B$. Thus, A belongs to $\mathcal{C}$. □

Recall that most average-case complexity classes discussed in this thesis enjoy the sparse interpolation property. In particular, we have:

Proposition 7.2.17 **[97]** *Let $\mathcal{F}$ be a set of distributions with $\nu_{\mathrm{tally}} \in \mathcal{F}$. For every $\mathcal{C} \in \{\mathbf{\Delta}_k^{\mathrm{p}}, \mathbf{\Sigma}_k^{\mathrm{p}}, \mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}, \mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}}, \mathbf{BPP}, \mathbf{PSPACE} \mid k > 0\}$,* TALLY $\cap\, \mathcal{C}_{\mathcal{F}} \subseteq \mathcal{C}$.

Several corollaries follow from Proposition 7.2.17.

Corollary 7.2.18 *Let $k > 0$. For any set $\mathcal{F}$ of distributions, $\mathbf{\Delta}_k^{\mathrm{e}} \not\subseteq \mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}}$, $\mathbf{\Sigma}_k^{\mathrm{e}} \not\subseteq \mathbf{\Sigma}_{k\mathcal{F}}^{\mathrm{p}}$, and $\mathbf{\Pi}_k^{\mathrm{e}} \not\subseteq \mathbf{\Pi}_{k\mathcal{F}}^{\mathrm{p}}$.*

Proof. The claim follows from the fact that TALLY $\cap\, \mathbf{\Delta}_k^{\mathrm{e}} \not\subseteq \mathbf{\Delta}_k^{\mathrm{p}}$ but TALLY $\cap\, \mathbf{\Delta}_{\mathcal{F}}^{\mathrm{p}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}$. The cases for $\mathbf{\Sigma}_{k\mathcal{F}}^{\mathrm{p}}$ and $\mathbf{\Pi}_{k\mathcal{F}}^{\mathrm{p}}$ are similar. □

Corollary 7.2.19 *Let $\mathcal{F}$ be a set of distributions with ν_{tally}. For each $k > 0$, if $\mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}}$, then $\mathbf{\Delta}_k^{\mathrm{e}} = \mathbf{\Sigma}_k^{\mathrm{e}}$.*

Proof. Assume that $\mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}}$. By Proposition 7.2.17, TALLY $\cap\, \mathbf{\Sigma}_k^{\mathrm{p}} \subseteq$ TALLY $\cap\, \mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}$, and thus TALLY $\cap\, \mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}$. This is equivalent to $\mathbf{\Delta}_k^{\mathrm{e}} = \mathbf{\Sigma}_k^{\mathrm{e}}$. □

Corollary 7.2.20 *Let $k \geq 1$ and let $\mathcal{F}$ be a set of distributions with ν_{tally}. If $\mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}} = \mathbf{\Sigma}_{k\mathcal{F}}^{\mathrm{p}}$, then $\mathbf{\Delta}_k^{\mathrm{e}} = \mathbf{\Sigma}_k^{\mathrm{e}}$.*

Proof. The claim is another variant of Corollary 7.2.19 because $\mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}} = \mathbf{\Sigma}_{k\mathcal{F}}^{\mathrm{p}}$ implies $\mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}}$. □

It is unlikely that $\mathbf{\Delta}_k^{\mathrm{e}} = \mathbf{\Sigma}_k^{\mathrm{e}}$; thus we may conjecture that $\mathbf{\Sigma}_k^{\mathrm{p}} \not\subseteq \mathbf{\Delta}_{k\mathcal{F}}^{\mathrm{p}}$ for all $\mathcal{F}$ which contains ν_{tally}. In other words, the sets in $\mathbf{\Sigma}_k^{\mathrm{p}}$ seem hard to compute even on average.

When we consider the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$, the claim of Proposition 7.2.17 can be strengthened in the following manner. Recall that a set S is $\mathbf{P}$-printable if and only if there exists a polynomial-time computable function which, on input 0^n, lists all strings in S of length n.

Lemma 7.2.21 **[98]** *Let A be an arbitrary set. If A is in $\mathbf{P}_{\mathbf{P}\text{-comp}}$, then $A \cap S$ is in $\mathbf{P}$ for any $\mathbf{P}$-printable set S. If $\mathbf{P} = \mathbf{NP}$ holds, then this is also true for all sparse sets S in $\mathbf{P}$.*

Proof. Let A be an arbitrary set in $\mathbf{P}_{\mathbf{P}\text{-comp}}$. Take any $\mathbf{P}$-printable set S. Under our assumption, we may assume that there exists an integer $k > 0$ such that $\|S \cap \Sigma^n\| = (n+1)^k$ for all $n \in \mathbb{N}$, since if the density of $S \cap \Sigma^n$ does not reach $(n+1)^k$ then we can deterministically add an element, which is not in S,

to S repeatedly until its density is exactly $(n+1)^k$.

Let us define $\hat{\mu}(x) \propto (|x|+1)^{-k-2}$ if $x \in S$; else 0. Notice that μ is $\mathbf{P}$-computable. Take a machine M which computes A in polynomial time on μ-average. There are integers $m, c > 0$ such that $\hat{\mu}(\{x \mid \mathrm{Time}_M(x) > c(|x| \cdot r)\}) < 1/r$ for all $r > 0$. For any fixed $x \in A \cap S$, we have $\mathrm{Time}_M(x) \leq c(|x|/\hat{\mu}(x))^m \leq c'|x|^m(|x|+1)^{m(k+2)}$. Hence, $A \cap S \in \mathbf{P}$.

The latter half of the claim follows from the fact that all sparse sets in $\mathbf{P}$ are $\mathbf{P}$-printable if $\mathbf{P} = \mathbf{NP}$. □

We have seen that the tally part of quintessentially computable class $\mathcal{C}_{\mathcal{F}}$ is easily computed, and thus it collapses to its worst-case counterpart $\mathcal{C}$. If we take the set of recursive distributions $\mathbf{REC}$-comp, then $\mathcal{C}_{\mathbf{REC}\text{-comp}}$ collapses to $\mathcal{C}$. To prove this, we show that if $\mathrm{Aver}(\mathcal{C}, \mathbf{REC}\text{-comp})$ has the sparse interpolation property, then $\mathcal{C}_{\mathbf{REC}\text{-comp}} \subseteq \mathcal{C}$. In the proof of the following lemma, we again use infinite, recursive, proper hard cores (see Definition 2.5.11).

Lemma 7.2.22 **[97]** *Let* $\mathrm{Aver}(\mathcal{C}, \mathbf{REC}\text{-comp})$ *be an average-case complexity class with* $\mathcal{C} \subseteq \mathbf{REC}$. *If* $\mathrm{Aver}(\mathcal{C}, \mathbf{REC}\text{-comp})$ *has the sparse interpolation property, then* $\mathcal{C}_{\mathbf{REC}\text{-comp}} \subseteq \mathcal{C}$.

Proof. Suppose that $\mathrm{Aver}(\mathcal{C}, \mathbf{REC}\text{-comp})$ has the sparse interpolation property. The proof is by contradiction. Now assume that there exists a set A in $\mathcal{C}_{\mathbf{REC}\text{-comp}} - \mathcal{C}$. By Lemma 2.5.12, there exists an infinite, recursive, proper hard core H for A with respect to $\mathcal{C}$. We note that if $\mathcal{C} = \mathbf{P}$, then H is in the class $\mathbf{E}$ (see, e.g., [4]). Thus, for any set $B \in \mathcal{C}$, if $B \subseteq A$, then $B \cap H$ is finite. Now let S be a recursive, infinite, sparse subset of H. Let $q(n) = \|S \cap \Sigma^n\|$. Consider the distribution $\mu_{S,q}$ such that

$$\hat{\mu}_{S,q}(x) = \begin{cases} \frac{\hat{\nu}_{\mathrm{tally}}(1^{|x|})}{q(|x|)} & \text{if } x \in S - \{\lambda\}, \\ 0 & \text{if } x \notin S \cup \{\lambda\}, \\ 1 - \sum_{z : z \neq \lambda} \hat{\mu}_{S,q}(z) & \text{if } x = \lambda. \end{cases}$$

Clearly $\mu_{S,q} \in \mathbf{REC}$-comp. Since $(A, \mu_{S,q}) \in \mathrm{Aver}(\mathcal{C}, \mathbf{REC}\text{-comp})$, there exists an interpolant $B' \in \mathcal{C}$ of A and S. We then have $B' \cap H \supseteq S$, and thus $B' \cap H$ is infinite. This contradicts the fact that H is a proper hard core for A. □

Theorem 7.2.23 **[97]** *Let* $k > 0$.

1. $\boldsymbol{\Delta}_k^{\mathrm{p}}\mathbf{REC}\text{-comp} = \mathbf{A}\boldsymbol{\Delta}_k^{\mathrm{p}}\mathbf{REC}\text{-comp} = \boldsymbol{\Delta}_k^{\mathrm{p}}$.
2. $\boldsymbol{\Sigma}_k^{\mathrm{p}}\mathbf{REC}\text{-comp} = \mathbf{A}\boldsymbol{\Sigma}_k^{\mathrm{p}}\mathbf{REC}\text{-comp} = \boldsymbol{\Sigma}_k^{\mathrm{p}}$.
3. $\mathbf{BPP}_{\mathbf{REC}\text{-comp}} = \mathbf{BPP}$.
4. $\mathbf{PSPACE}_{\mathbf{REC}\text{-comp}} = \mathbf{PSPACE}$.

Proof. By Lemma 7.2.22, it suffices to show that, for $\mathcal{C} \in \{\boldsymbol{\Delta}_k^{\mathrm{p}}, \boldsymbol{\Sigma}_k^{\mathrm{p}}, \mathbf{BPP}, \mathbf{PSPACE}\}$, $\mathrm{Aver}(\mathcal{C}, \mathbf{REC}\text{-comp})$

has the sparse interpolation property. The claim for $\mathcal{C} \in \{\mathbf{BPP}, \mathbf{PSPACE}\}$ follows from Proposition 6.4.9, and the claim for $\mathcal{C} \in \{\mathbf{\Delta}_k^{\mathrm{p}}, \mathbf{\Sigma}_k^{\mathrm{p}} \mid k > 0\}$ follows from Proposition 6.4.11. □

Theorem 7.2.23 indicates that the definition of the average polynomial-time hierarchy in Section 6.4 is a reasonable generalization of the worst-case polynomial-time hierarchy.

Note that, in the proof of Lemma 7.2.22, the complexity of the distribution $\mu_{S,q}$ depends only on the complexity of the complexity core. Since all sets not in $\mathbf{P}$ have complexity cores in $\mathbf{E}$, we get the following corollary.

Corollary 7.2.24 **[97]** $\mathbf{P}_{\mathbf{E}\text{-comp}} = \mathbf{P}$.

Complex distributions like recursive distributions make quintessential complexity classes lose their average-case nature. This supports our primary interests in feasible distributions in Chapter 4. In later sections, we shall focus on quintessential complexity classes under sets of those feasible distributions.

7.3 Fundamental Separations

We have seen the collapse of the real polynomial-time hierarchy under the set of recursive distributions. This section shows the separations between the real polynomial-time hierarchy under $\mathbf{P}$-computable distributions and the polynomial-time hierarchy. The technique cultivated in this section is fundamental and will be used again in later sections.

7.3.1 Construction of Hard Instances

Three years after the notion $\mathbf{P}_{\mathbf{P}\text{-comp}}$ was presented, Schuler [92] succeeded in showing that $\mathbf{P}_{\mathbf{P}\text{-comp}} \neq \mathbf{P}$ by constructing a complex set which lies in the difference $\mathbf{P}_{\mathbf{P}\text{-comp}} - \mathbf{P}$. A crucial idea in his proof is to find a string of each length which occurs with low probability by pruning other strings which occur with relatively high probability. The construction needs an effective enumeration of $\mathbf{P}$-computable semi-distributions. Later Schuler and Yamakami [98] extended this result to create sets which are hard to compute even by $O(2^{c \cdot n})$-time bounded Turing machines for a fixed constant $c > 0$. Certainly we cannot extend this result to $2^{O(n)}$-time bounded machines because all sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ are already in $\mathbf{E}$. Hence, this result seems nearly optimal.

The following lemma will be useful in later subsections. The proof given here uses Lemma 4.2.15, due to Schuler [93], which uses resource-bounded Kolmogorov complexity to avoid any enumeration of semi-distributions for the construction of a hard set.

For a set A, define $Prefix(A) = \{0^n 1 v \mid \exists w[|w| = n \wedge vw \in A]\}$.

Lemma 7.3.1 **(Hard Instance Lemma) [92, 98, 93]** *Let $k \geq 1$. Let $k(n)$ and $s(n)$ be time-constructible functions on $\mathbb{N}$. Assume that $k(n)$ and $s(n)$ are unbounded, increasing, and $1 \leq k(n) \leq n$ and $k(n) \in \Omega(n)$. Then, there exist a set $A \in \mathrm{DTIME}(n^{s(n)})$ and a function h computable in time $O(n^{s(n)})$*

such that, for all integers $n > 0$,

1. $|h(1^n)| = k(n)$;
2. $A \cap \Sigma^n = z\Sigma^{n-|z|}$, *where* $z = h(1^n)$;
3. $A \cap B \in \mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ *for every set* $B \in \mathbf{\Delta}^{\mathrm{e}}_k$;
4. $A \cap B \in \mathbf{A}\mathbf{\Sigma}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ *for every set* $B \in \mathbf{\Sigma}^{\mathrm{e}}_k$; *and*
5. $Prefix(A) \in \mathbf{P}_{\mathbf{P}\text{-comp}}$.

Proof. Without loss of generality, we may assume that $s(n) \leq \log n$ for all n. Since $k(n) \in \Omega(n)$, we can take an integer $c_1 > 0$ such that $k(n) \geq \frac{n}{c_1}$ for all $n \in \mathbb{N}$. Let $l(1) = k'(1) = 1$. For $n \geq 2$, let $l(n) = \lfloor 10 \log n \rfloor$ and let $k'(n) = \max\{1, \lfloor k(n)/10 \cdot \log n \rfloor\}$. Now consider an integer $n_1 > 0$ such that $n \geq 40c_1 \cdot \log n$ for all $n \geq n_1$. Obviously, $k'(n) \cdot l(n) \leq k(n)$ for all $n > 0$. Note that, for all $n \geq n_1$, $k'(n) \cdot l(n) \geq \frac{1}{2}k(n)$ since $k(n) \geq 40 \log n$ and, in consequence,

$$\begin{aligned} k'(n)l(n) &\geq \left(\frac{k(n)}{10\log n} - 1\right)(10\log n - 1) \geq k(n) - \left(\frac{k(n)}{10\log n} + 10\log n\right) \\ &\geq k(n) - \frac{1}{2}k(n) = \frac{1}{2}k(n). \end{aligned}$$

It is also clear that $\lambda x.l(|x|)$, $\lambda x.k(|x|)$, and $\lambda x.k'(|x|)$ are unbounded and computable in polynomial time since $k(n)$, $s(n)$, and $\lambda n.\lfloor 10 \log n \rfloor$ are time-constructible.

To simplify the following analysis, we always disregard the computation time for the values $k(n)$, $l(n)$, and $k'(n)$ because, as we have seen, $\lambda x.k(|x|)$, $\lambda x.s(|x|)$, and $\lambda x.l(|x|)$ are all $\mathbf{P}$-computable and do not affect "average" running time.

As for the desired function h, we define $h(1^n)$ to be a string $z^n_1 \cdots z^n_{k'(n)} 0^{k(n)-k'(n)l(n)}$, where $z^n_1 = \min\{w \in \Sigma^{l(n)} \mid w \notin \mathrm{KT}[n, 2^{s(1)\cdot n}]\}$ and $z^n_i = \min\{w \in \Sigma^{l(n)} \mid w \notin \mathrm{KT}[n, 2^{s(i)\cdot n}|z^n_1 \cdots z^n_{i-1}]\}$ for all i with $1 < i \leq k'(n)$. Obviously, we have $|h(1^n)| = k(n)$ for all $n > 0$. Now define the desired set A as $A = \{zw \mid \exists n[z = f(1^n) \wedge w \in \Sigma^{n-|z|} \wedge n > 0]\}$. Note that $A \cap \Sigma^n = h(1^n)\Sigma^{n-|h(1^n)|}$, and thus $\|A \cap \Sigma^n\| = 2^{n-k(n)}$ for all $n > 0$. We next show that $A \cap B \in \mathbf{A}\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ for all sets B in $\mathbf{\Delta}^{\mathrm{e}}_k$. Now fix a set B in $\mathbf{\Delta}^{\mathrm{e}}_k$. Consider the following algorithm M:

begin algorithm M for $A \cap B$
  input x (let $n = |x|$)
  compute $k(n), s(n), l(n)$, and $k'(n)$ (in polynomial time)
  let $y = \lambda$
  for $i = 1$ **to** $k'(n)$ **do**
    compute $z^n_i = \min\{w \in \Sigma^{l(n)} \mid w \notin \mathrm{KT}[n, 2^{s(i)\cdot n}|y]\}$
    let $y = yz^n_i$
    if $y \not\sqsubseteq x$ **then** reject
  end-for

if $y0^{k(n)-k'(n)l(n)} \not\sqsubseteq x$ **then** reject

(∗) **if** $x \in B$ **then** accept **else** reject

end.

It is easy to see that the algorithm M computes $A \cap B$. For later convenience, we call a string which goes into the line (∗) a *candidate*.

Now let μ be any **P**-computable distribution. By Lemma 4.2.15, there exists a constant $c_0, n_0 > 0$ such that, for all i with $c_0 < i \leq k'(n)$, $\hat{\mu}(z_1^n \cdots z_i^n \Sigma^{n-i\cdot l(n)}) < 2^{-(i-c_0)l(n)}$. Let c' be a large integer so that, in the following estimation of the time complexity of the algorithm, the inequalities (∗∗) and (∗ ∗ ∗) below hold for any integer $n > 0$. Moreover, we assume that the computation time of the set B on input of length n is at most $2^{c'\cdot n+c'}$ for each $n \in \mathbb{N}$. Let n_2 be a constant large enough so that $n_2 > \max\{n_0, n_1\}$, $n \geq 2c'$, and $n \geq 2c_1(10c_0+1)\log n + 1$ for all $n \geq n_1$. In what follows, we assume that n is any integer larger than n_2.

Let Z_i be the set of strings of length n which are not rejected through the first i iterations of the **for**-loop; namely, $Z_i = z_1^n \cdots z_i^n \Sigma^{n-i\cdot l(n)}$. By definition, $Z_0 = \Sigma^n$ and $A \subseteq Z_{k'(n)}$. Recall that $\hat{\mu}(Z_i) < 2^{-(i-c_0)l(n)}$. We then partition Σ^n into $k'(n)+2$ subsets, $\mathcal{S} = \{Z_{i-1} - Z_i \mid 1 \leq i \leq k'(n)\} \cup \{Z_{k'(n)} - A\} \cup \{A\}$. Note that $k'(n)+2 \leq n$ since $n \geq n_2$. By Lemma 3.3.15, it suffices to show that, for some constant $c > 0$ independent of the choice of n, $\mathrm{Time}_M(x) \leq n^c + (1/n^2\hat{\mu}(D))^c$ for all $x \in D$, where D is an arbitrary set in $\mathcal{S}$.

Note that, to compute z_i^n, in the worst case we have to check all strings w in $\Sigma^{l(n)}$ whether $w \notin \mathrm{KT}[n, 2^{s(i)\cdot n} \mid y]$. Hence, we need at most $2^{l(n)} \cdot 2^{s(j)l(n)} \leq 2^{(\log j+1)l(n)}$ steps because $\log j \geq s(j)$. Since $i \cdot l(n) + 1 \leq 2^{l(n)}$, the running time of M on input x which is in $Z_{i-1} - Z_i$ requires

$$\begin{aligned}
\mathrm{Time}_M(x) &\leq c' \sum_{j=1}^{i} (2^{(\log j+1)l(n)} + j\cdot l(n) + 1) \qquad (**)\\
&\leq c' \cdot i \cdot (2^{(\log i+1)l(n)} + 2^{l(n)})\\
&\leq 2^{\log c' + \log i} \cdot 2^{(\log i+1)l(n)+1} = 2^{\log c'' + \log i + 1} \cdot 2^{(\log i+1)l(n)}\\
&\leq 2^{2\log n} \cdot 2^{(\log i+1)l(n)}\\
&\leq 2^{l(n)+(\log i+1)l(n)} \leq 2^{(\log i+2)l(n)}
\end{aligned}$$

since $\log n \geq \log c' + 1$, $\log n \geq \log i$, and $l(n) \geq 2\log n$.

Let d be the minimal integer satisfying the following condition: $i \geq \log i + c_0 + 3$ for all integers $i > d$. Note that d does not depend on the value of n. Let $c = \max\{4c_1c', \lceil 5(\log d + 2)\rceil\}$. We examine below the running time of M for several cases of input string x.

(i) For the computation on all rejected strings x in $\Sigma^n - Z_d$, for $i \leq d$,

$$\mathrm{Time}_M(x) \leq 2^{(\log d+2)l(n)} \leq 2^{5(\log d+2)\log n} = n^{5(\log d+2)} \leq n^c.$$

(ii) For the computation on all rejected strings x in $Z_{i-1} - Z_i$ with $d < i \leq k'(n)$, as we have seen, $\mathrm{Time}_M(x) \leq 2^{(\log i+2)l(n)}$. Note that, since $i \geq \log i + c_0 + 3$, $(\log i + 2)l(n) \leq (i - c_0)l(n) - 2\log n$. Then we have

$$\mathrm{Time}_M(x) \leq 2^{(\log i+2)l(n)} \leq \frac{1}{n^2} \cdot 2^{(i-c_0)l(n)} \leq \frac{1}{n^2 \cdot \hat{\mu}(Z_{i-1} - Z_i)}.$$

(iii) For the computation on all rejected strings x in $Z_{k'(n)} - A$, note that $z_1^n \cdots z_{k'(n)}^n \sqsubseteq x$. We also note that $k(n) - k'(n)l(n) - 1 \leq 2^{\log k(n)+1}$. Hence, the time spent for x is

$$\begin{aligned} \mathrm{Time}_M(x) &\leq 2^{(\log k'(n)+2)l(n)} + c'(k(n) - k'(n)l(n) + 1) \qquad (***) \\ &\leq 2^{(\log k'(n)+2)l(n)} + 2^{\log c'} \cdot 2^{\log k(n)+1} \\ &\leq 2^{(\log k'(n)+2)l(n)+2\log n} \leq 2^{(\log k'(n)+3)l(n)} \end{aligned}$$

since $\log n \geq \log c' + 1$ and $l(n) \geq 2\log n$. Using the fact that $\hat{\mu}(Z_{k'(n)} - A) < 2^{-(k'(n)-c_0)l(n)}$, we have

$$\mathrm{Time}_M(x) \leq 2^{(\log k'(n)+3)l(n)} \leq 2^{(k'(n)-c_0)l(n)-2\log n} \leq \frac{1}{n^2 \cdot \hat{\mu}(Z_{k'(n)} - A)}.$$

(iv) For the computation on all strings x in A, we remark that the computation of B needs, by our assumption, at most $2^{c'\cdot n+c'}$ time. Hence, $\mathrm{Time}_M(x) \leq 2^{c'n+c'}$. Note that $\hat{\mu}(A) \leq \hat{\mu}(Z_{k'(n)}) < 2^{-(k'(n)-c_0)l(n)}$. Recall that $k'(n)l(n) \geq \frac{1}{2}k(n)$ and $k(n) \geq \frac{n}{c_1}$. We first remark that $c'n + c' \leq c((k'(n)l(n) - c_0)l(n) - 2\log n)$. This is seen as follows: since $c \geq 4c'c_1$ and $n \geq 2c_1(10c_0 + 1)\log n + 1$,

$$\begin{aligned} \frac{c'n + c'}{c} - (k'(n) - c_0)l(n) &\leq \frac{n+1}{4c_1} - \frac{k(n)}{2} - c_0 \cdot l(n) \\ &\leq \frac{n+1}{4c_1} - \frac{n}{2c_1} + 10c_0 \cdot \log n \\ &\leq -\frac{n}{2c_1} + 10c_0 \cdot \log n + \frac{1}{4c_1} \leq -2\log n. \end{aligned}$$

Therefore, we have

$$\mathrm{Time}_M(x) \leq 2^{c'n+c'} \leq 2^{c((k'(n)-c_0)l(n)-2\log n)} \leq \left(\frac{1}{n^2 \cdot \hat{\mu}(A)}\right)^c.$$

By Lemma 3.3.15, we conclude that $\lambda x.\mathrm{Time}_M(x)$ is polynomial on μ-average. Since μ is arbitrary, $A \cap B$ belongs to $\mathbf{A}\boldsymbol{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp}$.

The same argument can be carried out for the claim (4).

Next we show that claim (5) also holds. To prove (5), we consider the following algorithm M'.

begin algorithm M' for $Prefix(A)$
  input x
  find v such that $x = 0^{n-|v|}1v$
  if no such v exists **then** reject
  let $y = \lambda$
  for $i = 1$ **to** $k'(n)$ **do**
    compute $z_i^n = \min\{w \in \Sigma^{l(n)} \mid w \notin \mathrm{KT}[n, 2^{s(i)\cdot n}|y]\}$
    let $y = yz_i^n$
    if $v \sqsubseteq y$ **then** accept
    if $y \not\sqsubseteq v$ **then** reject
  end-for
  if $v \sqsubseteq y0^{k(n)-k'(n)l(n)}$ **then** accept

if $y0^{k(n)-k'(n)l(n)} \not\sqsubseteq v$ **then** reject **else** accept

end.

Fix n sufficiently large. Let $V_i^n = \{0^{n-|v|}1v \mid (i-1)l(n) < |v| \leq i \cdot l(n) \wedge v \sqsubseteq z_1^n \cdots z_i^n\}$ for each i, $1 \leq i \leq k'(n)$. For the sake of convenience, let $V_0^n = \Sigma^{n+1} - \bigcup_{i=1}^{k'(n)} V_i^n$. Notice that $\Sigma^{n+1} = \bigcup_{i=0}^{k'(n)} V_i^n$.

Take an arbitrary $\mathbf{P}$-computable distribution ν. It suffices to show that, for each i $(1 \leq i \leq k'(n))$, $\mathrm{Time}_{M'}(x) \leq n^c + (1/(n+1)^2\hat{\nu}(V_i^n))$ holds for all $x \in V_i^n$. First we show that $\hat{\nu}(V_i^n) < 2^{-(i-c_0)\cdot l(n)}$, where c_0 is a constant chosen above.

Claim 17 $\hat{\nu}(V_i^n) < 2^{-(i-c_0)\cdot l(n)}$.

Proof of claim. Let us define f as follows. For each x,

$$f(x) = \begin{cases} z_i^n \cdots z_i^n 0^{n-i\cdot l(n)} & \text{if } x = 0^{n-|v|}1v \text{ for some } v \\ 0^n & \text{otherwise,} \end{cases}$$

where $i = \min\{i' \mid (i'-1)l(n) < |v| \leq i' \cdot l(n)\}$. The function f is increasing and $\mathbf{P}$-computable. Furthermore, we obtain $f(V_i^n) \subseteq Z_i^n$, and as a consequence, $V_i^n \subseteq f^{-1}(Z_i^n)$. For ν, we set $\mu = \nu_{f^{-1}}$. As discussed above, $\hat{\mu}(Z_i^n) < 2^{-(i-c_0)l(n)}$ holds. This yields the estimation of $\hat{\nu}(V_i^n)$ as follows.

$$\hat{\nu}(V_i^n) \leq \hat{\nu}(f^{-1}(Z_i^n)) = \hat{\mu}(Z_i^n) < 2^{-(i-c_0)\cdot l(n)}.$$

Therefore, it follows that $\hat{\nu}(V_i^n) < 2^{-(i-c_0)\cdot l(n)}$. ■

Using an argument analogous to the proof for (1), we can prove that $\mathrm{Time}_N(x) \leq n^c + (1/(n+1)\cdot\hat{\nu}(V_i^n))^c$ for some constant c. This completes the proof. □

7.3.2 Separation from "Quasi" Linear Exponential Time

This subsection will apply the Hard Instance Lemma to show that $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ contain sets which are hard to compute. In particular, we shall show the separation of the classes $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ from $\mathrm{ATIME}^{\Delta}(k, O(2^{c\cdot n}))$ and $\mathrm{ATIME}^{\Sigma}(k, O(2^{c\cdot n}))$, respectively.

We have seen that all tally sets in $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ ($\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$, resp.) collapse to $\mathbf{\Delta}_k^{\mathrm{p}}$ ($\mathbf{\Sigma}_k^{\mathrm{p}}$, resp.). We first show the existence of sparse sets in $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ and in $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$, respectively, which do not collapse to $\mathbf{\Delta}_k^{\mathrm{p}}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}$, i.e., $\mathrm{SPARSE} \cap \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{\Delta}_k^{\mathrm{p}}$ and $\mathrm{SPARSE} \cap \mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{\Sigma}_k^{\mathrm{p}}$.

To help the reader understand the following proofs, we first consider a strategy for proving $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{P}$. The basic idea used in this section is a traditional *diagonalization technique*. The Hard Instance Lemma guarantees the existence of a hard set A whose intersection with any set in $\mathbf{E}$ falls into $\mathbf{P}_{\mathbf{P}\text{-comp}}$. Now take a set D "diagonalized" against all sets in $\mathbf{P}$. Since D is in $\mathbf{E}$, we simply consider the intersection $A \cap D$. This intersection still belongs to $\mathbf{P}_{\mathbf{P}\text{-comp}}$ but does not belong to $\mathbf{P}$.

Now we are ready for a general theorem.

Theorem 7.3.2 *Let $k > 1$ and let c be any positive real number.*

1. $\mathrm{SPARSE} \cap \mathbf{A}\boldsymbol{\Delta}^{\mathrm{p}}_{k}\mathbf{P}\text{-comp} \not\subseteq \mathrm{ATIME}^{\Delta}(k, O(2^{c \cdot n}))$.

2. $\mathrm{SPARSE} \cap \mathbf{A}\boldsymbol{\Sigma}^{\mathrm{p}}_{k}\mathbf{P}\text{-comp} \not\subseteq \mathrm{ATIME}^{\Sigma}(k, O(2^{c \cdot n}))$.

Proof. Let c be fixed.

(1) Let $\{M_i\}_{i \in \mathbb{N}}$ be an effective enumeration of all semi-deterministic alternating Turing machines which work with at most k-alternations in time $O(2^{c \cdot n})$, with infinitely many repetitions. Now we define the diagonalized set $D = \{x \mid x \notin L(M_{\lfloor \log |x| \rfloor})\}$. It is not hard to see that the set D belongs to $\mathrm{ATIME}^{\Delta}(k, O(2^{(c+1)n}))$, but not to $\mathrm{ATIME}^{\Delta}(k, O(2^{c \cdot n}))$. By Lemma 7.3.1, we have a set A of density 1 such that $A \cap D \in \mathbf{A}\boldsymbol{\Delta}^{\mathrm{p}}_{k}\mathbf{P}\text{-comp}$. We show that $A \cap D \notin \mathrm{ATIME}^{\Delta}(k, O(2^{c \cdot n}))$. Assume that there exists the ith machine M_i which computes $A \cap D$. Let x be any string x of length 2^i. Then,

$$x \in A \cap D \Longleftrightarrow x \in D \Longleftrightarrow x \notin L(M_i) \Longleftrightarrow x \notin A \cap D.$$

This is a contradiction. Thus, $A \cap D \notin \mathrm{ATIME}^{\Delta}(k, O(2^{cn}))$.

(2) Consider a set A and a function h as defined in Lemma 7.3.1 such that $A \cap \Sigma^n = \{h(1^n)\}$ for all $n > 0$. Recall that A belongs to $\mathrm{DTIME}(O(n^{\log^* n}))$. By an "alternation" version of a time hierarchy theorem [99], there is a tally set B in the difference $\mathrm{ATIME}^{\Sigma}(k, O(2^{cn} \cdot \log n)) - \mathrm{ATIME}^{\Sigma}(k, O(2^{cn}))$. Define $C = \{x \mid x \in A \wedge 0^{|x|} \in B\}$. Clearly $C \in \boldsymbol{\Sigma}^{\mathrm{p}}_{k}\mathbf{P}\text{-comp}$ by Lemma 7.3.1. Now we prove that C is not in $\mathrm{ATIME}^{\Sigma}(k, O(2^{cn}))$. Assume otherwise. Consider the following algorithm that works on $\{0\}^*$:

begin alternating Turing machine
  input 0^n
  compute $h(1^n)$
  if $h(1^n) \in C$ **then** accept **else** reject
end.

Since $A \cap \Sigma^n = \{h(1^n)\}$ for all n, the above algorithm computes A. The running time of the algorithm on input 0^n is

$$\begin{aligned} \mathrm{Time}_B(0^n) &\leq c' \cdot \sum_{z : |z| = n} \mathrm{Time}_A(z) + c' \cdot \mathrm{Time}_C(x_n) \\ &\leq c' \cdot 2^n \cdot n^{\log^* n} + c' \cdot 2^{cn} \leq 2c' \cdot 2^{cn} \end{aligned}$$

for any sufficiently large integer n, where c' is an appropriate positive constant. Therefore, we have $B \in \mathrm{ATIME}^{\Sigma}(k, O(2^{cn}))$. This is a contradiction. □

Corollary 7.3.3 **[97]** *For any constant $c > 0$, $\mathbf{P}\text{-comp} \not\subseteq \mathrm{DTIME}(O(2^{c \cdot n}))$.*

Now recall that $\mathbf{\Delta}_k^{\mathrm{e}} = \mathrm{ATIME}^{\Delta}(k, 2^{O(n)})$ and $\mathbf{\Sigma}_k^{\mathrm{e}} = \mathrm{ATIME}^{\Sigma}(k, 2^{O(n)})$. Since $\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp} \subseteq \mathbf{\Delta}_k^{\mathrm{e}}$ and $\mathbf{A}\mathbf{\Sigma}_k^{\mathrm{p}}\mathbf{P}\text{-comp} \subseteq \mathbf{\Sigma}_k^{\mathrm{e}}$, the results of Theorem 7.3.2 are nearly optimal; thus the classes $\mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}\mathbf{P}\text{-comp}$ share hard sets with $\mathbf{\Delta}_k^{\mathrm{e}}$ and $\mathbf{\Sigma}_k^{\mathrm{e}}$. As a particular case, the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ has hard sparse sets in $\mathbf{E}$. In light of resource-bounded measure theory, however, the class $\mathrm{DTIME}(O(2^{c\cdot n}))$ is known to be *small* within $\mathbf{E}$, and this suggests that the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ may not be a large class within $\mathbf{E}$. In the later section, we shall show that $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is actually *small*.

Clearly $\mathbf{P} \subseteq \mathbf{P}_{\mathbf{P}\text{-samp}} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$ since $\mathbf{P}$-comp $\subseteq \mathbf{P}$-samp. Can we show that $\mathbf{P}_{\mathbf{P}\text{-samp}} \neq \mathbf{P}$? In contrast to Theorem 7.3.2, if $\mathbf{P}_{\mathbf{P}\text{-samp}} \neq \mathbf{P}$, we can solve some open questions in worst-case complexity theory.

Proposition 7.3.4 *If* $\mathbf{P}_{\mathbf{P}\text{-samp}} \neq \mathbf{P}$, *then either* $\mathbf{FP}^{\mathbf{E}} \not\subseteq \#\mathbf{P}$ *or* $\mathbf{NP} \not\subseteq \mathbf{BPP}$ *holds.*

Proof. We shall prove the contrapositive. Assume that $\mathbf{FP}^{\mathbf{E}} \subseteq \#\mathbf{P}$ and $\mathbf{NP} \subseteq \mathbf{BPP}$. Since $\mathbf{P}$-samp $\subseteq$ $\#\mathbf{P}$-comp, we have $\mathbf{P}_{\#\mathbf{P}\text{-comp}} \subseteq \mathbf{P}_{\mathbf{P}\text{-samp}}$. Under the assumption $\mathbf{NP} \subseteq \mathbf{BPP}$, by Corollary 4.7.7, $\mathbf{P}_{\mathbf{P}\text{-samp}} = \mathbf{P}_{\#\mathbf{P}\text{-comp}}$. Our assumption $\mathbf{FP}^{\mathbf{E}} \subseteq \#\mathbf{P}$ leads to the conclusion that $\mathbf{E}$-comp $\subseteq \#\mathbf{P}$-comp since

$$\mathbf{E}\text{-comp} \subseteq \mathbf{P}^{\mathbf{E}}\text{-comp} \subseteq \#\mathbf{P}\text{-comp}.$$

Hence, we have $\mathbf{P}_{\#\mathbf{P}\text{-comp}} \subseteq \mathbf{P}_{\mathbf{E}\text{-comp}}$. Recall from Corollary 7.2.24 that $\mathbf{P}_{\mathbf{E}\text{-comp}} = \mathbf{P}$. This yields the desired conclusion that $\mathbf{P}_{\mathbf{P}\text{-samp}} = \mathbf{P}$. □

Here we show another application of Lemma 7.3.1.

Definition 7.3.5 (Sparsely Close Sets) For a complexity class $\mathcal{C}$, a set S is called *(sparsely)* $\mathcal{C}$*-close* if there exists a set $B \in \mathcal{C}$ such that $A \triangle B$ is sparse. For the sake of convenience, we also use the notation $\mathcal{C}$-close to denote the collection of all $\mathcal{C}$-close sets.

Note that any recursive set whose polynomial complexity core is sparse belongs to $\mathbf{P}$-close.

Proposition 7.3.6 *Let* $k > 0$. *For any* $c > 0$, $\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp} \not\subseteq \mathrm{ATIME}^{\Delta}(k, O(2^{c\cdot n}))$-close. *In particular,* $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{P}$-close.

Proof. The proof is by diagonalization. Let $C = \{x \mid x \notin L(M_{\lfloor \log |x| \rfloor})\}$, where $\{M_i\}_{i \in \mathbb{N}}$ is an effective enumeration of semi-deterministic k-alternation, $O(2^{cn})$-time bounded alternating Turing machines with infinitely many repetitions. We have $C \in \mathrm{ATIME}^{\Delta}(O(2^{(c+1)n}))$. Define A as in the Hard Instance Lemma by choosing $n - \lceil \log^2 n \rceil$ as $k(n)$. It follows from Lemma 7.3.1 that $A \cap C \in \mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp}$.

We next show that A is not sparse. Note that $\|(A \cap C) \triangle L(M_{\lfloor \log n \rfloor}) \cap \Sigma^n\| = \|A \cap \Sigma^n\|$. The density of A is, for each n,

$$\|A \cap \Sigma^n\| = 2^{\lceil \log^2 n \rceil} \geq 2^{\log^2 n} = n^{\log n}.$$

Hence, $(A \cap C) \triangle L(M_i)$, $i \in \mathbb{N}$, is not sparse since infinitely many machines coincide with M_i. □

Recall that **NT** is the collection of all near-testable sets. It is known that $\mathbf{P} \subseteq \mathbf{NT} \subseteq \mathbf{E} \cap \mathbf{PSPACE}$ [31]. We show that $\mathbf{P}_{\mathbf{P}\text{-comp}}$ contains a non near-testable set.

Proposition 7.3.7 **[98]** $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{NT}$.

Proof. Define A as in Theorem 7.3.2(1) with density given by $\|A \cap \Sigma^n\| = 1$ for each $n > 0$, so that A separates $\mathbf{P}_{\mathbf{P}\text{-comp}}$ from $\mathrm{DTIME}(O(2^n))$. We show that A is not near-testable.

Assume otherwise. Let $boundary(A) = \{x \mid x \neq \lambda$, exactly one of x and x^+ is in $A\}$, where x^+ is the successor of x. Notice that $boundary(A) = boundary(\overline{A})$. We claim that A is near-testable if and only if $boundary(A)$ is in $\mathbf{P}$. Define f by $f(x) \equiv \chi_A(x) + \chi_A(x^-) \pmod 2$ for all x. By the definition fo f, $f(x) = 1$ if and only if exactly one of x and x^- belongs to A. This yields the claim.

Thus, we have $boundary(A) \in \mathbf{P}$. Note that the set $A \cap \Sigma^n$ contains at most one string, i.e., $A \in \mathrm{SPARSE}$. Consequently, A is in $\mathbf{P}$. The conclusion that $A \in \mathbf{P}$ contradicts the fact that $A \notin \mathrm{DTIME}(O(2^n))$. □

However, it is open whether $\mathbf{P}_{\mathbf{P}\text{-comp}} \subseteq \oplus\mathbf{P}$ or not.

7.3.3 Separation from Advice Hierarchy

Another immediate consequence of the Hard Instance Lemma below is to show a relationship between $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ and the class $\mathbf{\Delta}_k^{\mathrm{p}}/cn$ defined by linear advice functions (see, e.g., [4, 41]). First we introduce an advise hierarchy in the following general fashion.

Definition 7.3.8 **($\mathcal{C}$ with Advice f)** For $\mathcal{C}$ a complexity class and f a function from $\mathbb{N}$ to Σ^*, a set S is in $\mathcal{C}/f$ ($\mathcal{C}$ with advice f) if there is a set $B \in \mathcal{C}$ such that $S \cap \Sigma^n = \{x \in \Sigma^n \mid \langle x, f(n)\rangle \in B\}$ for all integers $n \geq 0$. For a class $\mathcal{F}$ of functions from $\mathbb{N}$ to Σ^*, let $\mathcal{C}/\mathcal{F} = \bigcup_{f \in \mathcal{F}} \mathcal{C}/f$.

For example, the class $\mathbf{P}/\mathrm{poly}$ is known as the class of sets computable by non-uniform families of polynomial-size circuits. For the general properties of the advice hierarchy $\{\mathbf{\Delta}_k^{\mathrm{p}}/\mathrm{poly}, \mathbf{\Sigma}_k^{\mathrm{p}}/\mathrm{poly}, \mathbf{\Pi}_k^{\mathrm{p}}/\mathrm{poly} \mid k \in \mathbb{N}\}$, see Yap [121]. Note that whether $\mathbf{\Sigma}_k^{\mathrm{p}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}/\mathrm{poly}$ is an open question.

We want to show that $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is not included in $\mathbf{\Delta}_k^{\mathrm{p}}/cn$ when c is fixed.

Theorem 7.3.9 *Let $k > 0$. For each constant $c > 0$, $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{\Delta}_k^{\mathrm{p}}/cn$.*

Proof. Again we use a diagonalization argument. Let A be the set in $\mathrm{ATIME}^{\Delta}(k, O(n^{\log^* n}))$ whose existence is guaranteed by Lemma 7.3.1, such that $\|A^n\| = 2^{\lfloor \log n^2 \rfloor}$ and $A \cap B \in \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ for every set $B \in \mathbf{\Delta}_k^{\mathrm{e}}$.

Now let $\{M_i\}_{i \in \mathbb{N}}$ be an effective enumeration of all polynomial-time semi-deterministic k-alternation bounded alternating Turing machines, each of which, M_i, runs in $n^i + i$ steps. We shall define a set which diag-

onalizes against all such machines. For each machine M_i, let $ACC(D,x) = \{z \in D \mid M_{\lfloor \log |x| \rfloor} \text{ accepts } \langle x,z \rangle\}$ and $REJ(D,x) = \{z \in D \mid M_{\lfloor \log |x| \rfloor} \text{ rejects } \langle x,z \rangle\}$. Let us consider the following algorithm:

begin semi-deterministic alternating Turing machine
  input x (say, $|x| = n$)
  if $x \notin A$ **then** reject
  enumerate all elements in $A \cap \Sigma^n$ (say, $x_1, \ldots, x_m$, where $m = 2^{\lfloor \log n^2 \rfloor}$)
  (assume that $x = x_k$)
  let $D_1 = \Sigma^{cn}$
  if $k = 1$ **then go to** $(*)$
  for $i = 1$ **to** $k-1$ **do**
    if $D_i = \emptyset$ **then** reject
    if $\|ACC(D_i, x_i)\| > \|REJ(D_i, x_i)\|$ **then** $D_{i+1} = REJ(D_i, x_i)$
      else $D_{i+1} = ACC(D_i, x_i)$
  end-for
$(*)$ **if** $\|ACC(D_k, x_k)\| > \|REJ(D_k, x_k)\|$ **then** reject **else** accept
end.

Let B be the set accepted by the above Turing machine. Since $B \supseteq A$, it suffices to show that $B \in \mathbf{\Delta}_k^{\mathrm{e}}$. For each input x in A, the machine takes $O(2^{(c+1)n})$ steps since it needs $O(2^n \cdot n^{\log^* n})$ steps to enumerate all elements in $A \cap \Sigma^n$, at most n^2 iterations of the **for**-loop and $O(n^{\log n} \cdot 2^{cn})$ steps to compute each $ACC(D_i, x_i)$ and $REJ(D_i, x_i)$ for some constant $c > 0$. Therefore, B belongs to $\mathbf{\Delta}_k^{\mathrm{e}}$.

We show that $B \notin \mathbf{\Delta}_k^{\mathrm{p}}/cn$. Let us consider the set $\{D_i \mid 1 \leq i \leq 2^{\lfloor \log n^2 \rfloor}\}$. We remark that the definition of sets D_i, $1 \leq i \leq 2^{\lfloor \log n^2 \rfloor}$, does not depend on the choice of strings x in $A \cap \Sigma^n$. Consider the maximal k such that $D_k \neq \emptyset$, i.e., the algorithm goes into the line $(*)$. Note that k exists and $k \leq cn$. By our definition, either $ACC(D_k, x_k)$ or $REJ(D_k, x_k)$ is empty. By $(*)$, $ACC(D_k, x_k) \neq \emptyset$ exactly when $x_k \notin B$ holds. Hence, there is no advice string $z \in \Sigma^{cn}$ such that $x_i \in B$ if and only if $\langle x_i, z \rangle \in L(M_{\lfloor \log n \rfloor})$. □

Despite of the above theorem, it is still open whether $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}/\mathrm{poly}$. Schuler [93] presented negative evidence by demonstrating that if $\mathbf{P}_{\mathbf{P}\text{-comp}} \subseteq \mathbf{P}/\mathrm{poly}$, then $\mathbf{EXP} = \mathbf{\Sigma}_2^{\mathrm{p}}$. Here we wish to prove that $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}/\mathrm{poly}$ if and only if $\mathbf{\Delta}_k^{\exp} \subseteq \mathbf{\Delta}_k^{\mathrm{p}}/\mathrm{poly}$.

We shall show that every $\mathbf{\Delta}_k^{\exp}$-set is reducible to $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ via polynomial-size circuits. We first describe these reductions.

Definition 7.3.10 ($\mathbf{P}/\mathcal{F}$-m-reductions) Let h be a function from $\mathbb{N}$ to Σ^*. A set A is *$\mathbf{P}/h$-m-reducible to* a set B, denoted by $A \leq_m^{\mathbf{P}/h} B$, if there exists a function $g \in \mathbf{FP}$ such that $A = \{x \mid g(\langle x, h(|x|)\rangle) \in B\}$. For a set $\mathcal{F}$, A is *$\mathbf{P}/\mathcal{F}$-m-reducible to* B, denoted by $A \leq_m^{\mathbf{P}/\mathcal{F}} B$, if $A \leq_m^{\mathbf{P}/h} B$ for some $h \in \mathcal{F}$.

Note that $A \leq_m^{\rm p} B$ and $B \leq_m^{\mathbf{P}/O(n)} C$ imply $A \leq_m^{\mathbf{P}/\mathrm{poly}} C$, where poly is, as before, the set of p-bounded functions from $\mathbb{N}$ to Σ^*.

Lemma 7.3.11 *For each $k \in \mathbb{N}$, the classes $\mathbf{\Delta}_k^{\rm p}/\mathrm{poly}$ and $\mathbf{\Sigma}_k^{\rm p}/\mathrm{poly}$ are closed downward under $\mathbf{P}/\mathrm{poly}$-m-reductions.*

Proof. We shall prove the lemma only for $\mathbf{\Delta}_k^{\rm p}/\mathrm{poly}$. Assume that $A \leq_m^{\mathbf{P}/\mathrm{poly}} B$ and $B \in \mathbf{\Delta}_k^{\rm p}/\mathrm{poly}$. Since $A \leq_m^{\mathbf{P}/\mathrm{poly}} B$, there exists a function g in $\mathbf{FP}$ and a p-bounded function h from $\mathbb{N}$ to Σ^* such that $A = \{x \mid g(\langle x, h(|x|)\rangle \in B\}$. Also by our assumption that $B \in \mathbf{\Delta}_k^{\rm p}/\mathrm{poly}$, there are a set $D \in \mathbf{\Delta}_k^{\rm p}$ and a p-bounded function f from $\mathbb{N}$ to Σ^* satisfying that $B = \{x \mid \langle x, f(|x|)\rangle \in D\}$.

Combining those two set equations, it follows that

$$A = \{x \mid \langle g(\langle x, h(|x|)\rangle), f(|x|)\rangle \in D\}.$$

To show that A is in $\mathbf{\Delta}_k^{\rm p}/\mathrm{poly}$, we define the set E as $E = \{\langle x, y\rangle \mid \langle g(\langle x, (y)_0\rangle), (y)_1\rangle \in D\}$, where $(y)_0$ and $(y)_1$ are decodings of y satisfying $y = \langle (y)_0, (y)_1\rangle$. Let us set $q(n) = \langle h(n), f(n)\rangle$ for all natural numbers n. Clearly q is p-bounded because both h and f are so. The definitions of E and q yields the desired conclusion that $A = \{x \mid \langle x, q(|x|)\rangle \in D\}$, which implies that $A \in \mathbf{\Delta}_k^{\rm p}/\mathrm{poly}$. □

The following lemma generalizes the special case ($k = 1$) proven in [93].

Lemma 7.3.12 *Let $k > 0$. Every set in $\mathbf{\Delta}_k^{\rm exp}$ is $\mathbf{P}/\mathrm{poly}$-m-reducible to some set in $\mathbf{\Delta}_{k\,\mathbf{P}\text{-comp}}^{\rm p}$. A similar claim holds for $\mathbf{\Sigma}_k^{\rm exp}$ and $\mathbf{\Sigma}_{k\,\mathbf{P}\text{-comp}}^{\rm p}$.*

Proof. Let S be a set in $\mathbf{\Delta}_k^{\rm exp}$. Note that every $\mathbf{\Delta}_k^{\rm exp}$ set is p-m-reducible to some set in $\mathbf{\Delta}_k^{\rm e}$ by Lemma 2.5.10. Hence, there exists a set S' in $\mathbf{\Delta}_k^{\rm e}$ such that $S \leq_m^{\rm p} S'$. Let us define the set L to be the collection of all strings x such that $x = zy$ for some $y \in S'$ and z with $|z| = |y| + b$, where $b \equiv |x| \pmod 2$. Clearly L is in $\mathbf{\Delta}_k^{\rm e}$.

Take a set A and a function h defined in the Hard Instance Lemma with the condition $|h(1^n)| = \lceil n/2 \rceil$. Since $L \in \mathbf{\Delta}_k^{\rm e}$, the intersection $L \cap A$ lies in $\mathbf{\Delta}_{k\,\mathbf{P}\text{-comp}}^{\rm p}$. Now let $T = L \cap A$. It follows that, for all y of length n, $y \in S'$ if and only if $h(1^n)y \in T$. Hence, $S' \leq_m^{\mathbf{P}/O(n)} T$. Since $S \leq_m^{\rm p} S'$, we have $S \leq_m^{\mathbf{P}/\mathrm{poly}} T$. □

Using the above lemma, we can show the intractability of the classes in the real polynomial-time hierarchy under $\mathbf{P}$-comp.

Theorem 7.3.13 *Let $k \geq 1$. Let $\mathcal{C} \in \{\mathbf{\Delta}_k^{\rm p}, \mathbf{\Sigma}_k^{\rm p} \mid k \in \mathbb{N}\}$.*

1. *$\mathbf{\Delta}_{k\,\mathbf{P}\text{-comp}}^{\rm p} \subseteq \mathcal{C}/\mathrm{poly}$ if and only if $\mathbf{\Delta}_k^{\rm exp} \subseteq \mathcal{C}/\mathrm{poly}$.*

2. *$\mathbf{\Sigma}_{k\,\mathbf{P}\text{-comp}}^{\rm p} \subseteq \mathcal{C}/\mathrm{poly}$ if and only if $\mathbf{\Sigma}_k^{\rm exp} \subseteq \mathcal{C}/\mathrm{poly}$.*

7.4 Immunity and Bi-Immunity

Immune sets are another typical example of hard sets. In recursion theory, a set is called "immune" if it is infinite but contains no infinite recursive enumerable set. This notion has been adapted to complexity theory and used in a variety of situations.

Bi-immune sets are defined to be immune sets whose complements are also immune. These notions are fundamental and have already appeared elsewhere in this thesis. In this section, we shall pay more attention to the existence of such sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$.

7.4.1 Immune Sets and Complexity Cores

First of all, let us recall the formal definition of immunity. For a complexity class $\mathcal{C}$, a set S is *$\mathcal{C}$-immune* if S is infinite and S has no infinite subsets in $\mathcal{C}$.

A general notion of *complexity cores* has been introduced in Section 3.5, but in this subsection, we shall focus only on complexity cores with respect to $\mathbf{P}$, the so-called *polynomial complexity cores*. For a recursive set S, a set C is called a *polynomial complexity core for S* if, for any deterministic Turing machine M computing S and any polynomial p, the set $\{x \in C \mid \mathrm{Time}_M(x) \leq p(|x|)\}$ is finite.

This section will show that there exist $\mathbf{P}$-immune sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$, but $\mathbf{P}_{\mathbf{P}\text{-comp}}$ has no $\mathbf{P}$-bi-immune sets. Under the assumption that $\mathbf{P} = \mathbf{NP}$, all infinite polynomial complexity cores for sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ are shown to be hard to compute.

We first show the existence of a $\mathbf{P}$-immune set in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ of arbitrary density. The proof below uses an elegant technique developed by Ko and Moore [56].

Theorem 7.4.1 **[98]** *Let δ be a real number with $0 < \delta < 1$. There exists a $\mathbf{P}$-immune set in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ of density at least $2^{\delta n}$.*

Proof. Let $\{P_i\}_{i \in \mathbb{N}}$ be an effective enumeration of all sets in $\mathbf{P}$, where P_i is deterministically computed in $n^i + i$ steps.

Let $\epsilon = (1+\delta)/2$ and take the set A defined in the Hard Instance Lemma such that $\|A^n\| \geq 2^{\epsilon n}$. Note that $A \in \mathrm{DTIME}(O(n^{\log^* n}))$. Now consider the following algorithm:

```
begin
    input x (say, |x| = n)
    if x ∉ A then reject
    for i = 1 to log* n do
        if x ∉ P_i then go to (**)
        for all z (i ≤ |z| and z < x) do
            if z ∉ A then go to (*)
            if z ∈ P_i then go to (**)
(*)     end-for
```

```
                    accept
(**)          end-for
                    reject
          end.
```

Let B denote the set accepted by this algorithm and let $C = A - B$.

First, we see that C is of density at least $2^{\delta n}$. Assume $x, y \in B$ and $x \neq y$. Note that $x \in P_i$ and $y \in P_j$ are witnessed through the first **for**-loop, then $i \neq j$ by the algorithm. Hence, we have $\|B \cap \Sigma^n\| \leq \log^* n$. Since $\|A \cap \Sigma^n\| \geq 2^{\epsilon n}$, $\|C \cap \Sigma^n\| = \|A^n\| - \|B^n\| \geq 2^{\epsilon n} - \log^* n \geq 2^{\delta n}$.

We next show that C has no infinite $\mathbf{P}$-subset. Assume that an infinite set in $\mathbf{P}$ is a subset of C. Let i_0 be any integer that guarantees that P_{i_0} is such a set. Consider the minimal $x \in P_{i_0} \cap A_0$ such that $|x| \geq i_0$. By the minimality of x, we have $z \notin P_{i_0}$ for all z with $i_0 \leq |z|$ and $z < x$. Thus, $x \in B$, a contradiction. Therefore, C is $\mathbf{P}$-immune.

Now we claim that B is in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ since this implies $C \in \mathbf{P}_{\mathbf{P}\text{-comp}}$. Let $\mathrm{Time}_B(z)$ be the executing time of this algorithm on input z. It suffices to show that $\mathrm{Time}_B(x) \leq 2^{c'n}$ for all x in A, where c' is an absolute positive constant. Notice that $\mathrm{Time}_A(x) \leq c|x|^{2\log^* |x|}$ for some constant $c > 0$. Thus, the algorithm takes time $\sum_{z:|z|\leq n} \left[c|z|^{2\log^* |z|} + (|z|^i + i)\right]$ in the second **for**-loop, and this term is bounded by $O(n^{2\log^* n} \cdot 2^n) \subseteq O(2^{2n})$. The total execution time, $\mathrm{Time}_B(x)$, requires $\log^* n$ iterations of the first **for**-loop, each of which takes $O(2^{2n})$ steps, and therefore, $\mathrm{Time}_B(x) \leq 2^{c'n}$ for some constant $c' > 0$. □

We note that any $\mathbf{P}$-immune set is a polynomial complexity core for itself. Since the $\mathbf{P}$-immune set constructed in Theorem 7.4.1 is non-sparse, we immediately get the following corollary.

Corollary 7.4.2 **[98]** *There exists an infinite set in* $\mathbf{P}_{\mathbf{P}\text{-comp}}$ *which has a non-sparse polynomial complexity core in* $\mathbf{P}_{\mathbf{P}\text{-comp}}$.

Let us recall the complexity class **APT** introduced by Meyer and Paterson [71]. A set S is in **APT** if and only if the set $\{x \mid \mathrm{Time}_M(x) \geq p(|x|)\}$ is sparse for some polynomial p and some deterministic Turing machine M which computes A.

Corollary 7.4.3 **[98]** $\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{APT}$.

Proof. This result follows from Corollary 7.4.2 and the fact that a recursive set S is in **APT** if and only if any polynomial complexity core for S is sparse [78]. □

We have already seen the existence of *polynomially ℓ-rare* sets in Section 4.2. The particular example shown there was based on Kolmogorov complexity sets. Here we shall present another example of polynomially ℓ-rare sets based on complexity cores in $\mathbf{P}_{\mathbf{P}\text{-comp}}$.

Lemma 7.4.4 *Let ℓ be any positive function from $\mathbb{N}$ to $\mathbb{R}^+$. Let A be any set in* $\mathrm{DTIME}(2^{O(\ell(n)+\log n)}) \cap \mathbf{P}_{\mathbf{P}\text{-comp}}$. *Any complexity core for A with respect to* $\mathrm{DTIME}(2^{O(\ell(n)+\log n)})$ *is polynomially ℓ-rare.*

Proof. Let S be a complexity core for A with respect to $\mathrm{DTIME}(2^{O(\ell(n)+\log n)}) \cap \mathbf{P}_{\mathbf{P}\text{-comp}}$. To arrive at a contradiction, we assume that S is not polynomially ℓ-rare. From this assumption, it follows that there exists a $\mathbf{P}$-computable distribution μ satisfying the condition that the set $B = \{x \in S - \{\lambda\} \mid \hat{\mu}(x) > 2^{-\ell(n)}\}$ is infinite.

Since $A \in \mathbf{P}_{\mathbf{P}\text{-comp}}$, there exists a deterministic Turing machine M which computes A in polynomial time on μ-average. Let p be a polynomial such that $\lambda x.\mathrm{Time}_M(x)$ is p on μ-average. For any string x in B (say, $n = |x|$),

$$\mathrm{Time}_M(x) \leq p(2^{\ell(n)} \cdot n) \leq (2^{\ell(n)+\log n})^k$$

for some constant $k > 0$ independent of x. Now we set $B' = \{x \mid \mathrm{Time}_M(x) \leq (2^{\ell(n)+\log n})^k\}$. It is clear that B' belongs to $\mathrm{DTIME}(2^{O(\ell(n)+\log n)})$. Notice that $B' \cap S$ is infinite. This obviously contradicts our assumption that S is a complexity core for A. Therefore, S is polynomially l-rare. □

7.4.2 Bi-Immune Sets and Resource-Bounded Measure

We shall turn our interests to bi-immune sets. A set S is called *$\mathcal{C}$-bi-immune* if S and its complement $\overline{S}$ are both $\mathcal{C}$-immune. The class of $\mathbf{P}$-bi-immune sets has a close connection to resource-bounded measure theory. As seen in Proposition 2.7.11, any class which has no $\mathbf{P}$-bi-immune sets has p-measure 0; in other words, it is *small*.

It is known that the set $\mathbf{E}$ has "strongly" $\mathbf{P}$-bi-immune sets [5], and thus $\mathbf{E}$ has $\mathbf{P}$-bi-immune sets. It is important to remember here that a recursive set S is $\mathbf{P}$-bi-immune if and only if Σ^* is a polynomial complexity core for S. Hence, $\mathbf{E}$ contains a set for which Σ^* is a complexity core. However, we can see in the following proposition that there are no $\mathbf{P}$-bi-immune sets in $\mathbf{P}_{\mathbf{P}\text{-comp}}$. This contrast clearly shows the difference between $\mathbf{E}$ and $\mathbf{P}_{\mathbf{P}\text{-comp}}$.

Schuler and Yamakami [98] first showed that $\mathbf{P}_{\mathbf{P}\text{-comp}}$ has no $\mathbf{P}$-bi-immune sets. Later Schuler [94] extended their result as follows.

Proposition 7.4.5 **[94]** *Let $c > 0$. There are no* $\mathrm{DTIME}(O(2^{cn}))$*-bi-immune sets in the truth-table closure of* $\mathbf{P}_{\mathbf{P}\text{-comp}}$. *In particular, there are no* $\mathbf{P}$*-bi-immune sets in* $\mathbf{P}_{\mathbf{P}\text{-comp}}$.

Proof. Assume that B is $\mathrm{DTIME}(O(2^{cn}))$-bi-immune, and B is p-tt-reducible to a set A in $\mathbf{P}_{\mathbf{P}\text{-comp}}$. Without loss of generality, we assume that c is a positive integer. Let M be a polynomial-time oracle Turing machine which reduces B to A with nonadaptive queries. Let p be a polynomial such that $\mathrm{Time}_M(x) \leq p(|x|)$ for all x. Since $A \in \mathbf{P}_{\mathbf{P}\text{-comp}}$, there is a deterministic Turing machine N which computes A in time polynomial on μ-average for every $\mathbf{P}$-computable distribution μ. This shall lead to a contradiction.

We define a set D as follows. For each $n \in \mathbb{N}$, let $D_n = \{y \mid |y| \geq n/2c \wedge y \in Q(M, 0^n)\}$, and then

let $D = \{0\}^* \cup (\bigcup_{n>0} D_n)$. First we show that D is $\mathbf{P}$-printable. This is seen as follows. Consider the algorithm: on input 0^n, recursively take a natural number k such that $k \leq 2c \cdot n$, and list all strings in $\{y \in \Sigma^n \mid y \in Q(M, 0^k)\}$. Since M makes nonadaptive queries in polynomial time, this algorithm writes down all query strings of length n in polynomial time.

Now let us consider the set $A \cap D$. By Lemma 7.2.21, $A \cap D$ must be in $\mathbf{P}$. The following algorithm computes $B \cap \{0\}^*$:

begin algorithm for $B \cap \{0\}^*$
  input 0^n
  list all queries made by M on input 0^n
  for all query string y
    if $|y| \geq \frac{n}{2c}$ **then** set $ans(y) := [y \in A \cap D]$
(∗) **else** simulate N on input y and let $ans(y)$ be its output
  end-all
  simulate M on 0^n with oracle $\{y \mid ans(y) = 1\}$
  output $M(0^n)$
end.

In line (∗), the number of steps we need is at most $2^{c|y|} \leq 2^{c \cdot \frac{n}{2c}} = 2^{n/2}$. Hence, the total number of steps of this algorithm is, for some absolute constant $d > 0$,

$$d \cdot p(n) \cdot (p(n) + 2^{n/2} + 1) \leq (2d \cdot p(n)) \cdot 2^{n/2} \leq 2^{n/2} \cdot 2^{n/2} = 2^n$$

for every sufficiently large integer n. Therefore, the set $B \cap \{0\}^*$ belongs to DTIME($O(2^n)$). A similar argument shows that $\overline{B} \cap \{0\}^*$ is in DTIME($O(2^n)$).

Notice that at least one of the sets $B \cap \{0\}^*$ and $\overline{B} \cap \{0\}^*$ is infinite. This contradicts our assumption that B is DTIME($O(2^{c \cdot n})$)-bi-immune. □

For each constant $c > 0$, the class of DTIME($O(2^{c \cdot n})$)-bi-immune sets is known to have p-measure 1 [69]. In other words, the class of non-DTIME($O(2^{c \cdot n})$)-bi-immune sets has p-measure 0. Hence, by Propositions 7.4.5 and 2.7.11, the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ cannot have p-measure 1. Therefore, we obtain the following corollary given by Schuler [94].

Corollary 7.4.6 **[94]** *The truth-table closure of* $\mathbf{P}_{\mathbf{P}\text{-comp}}$ *has p-measure* 0*, and thus it has measure* 0 *in* $\mathbf{E}$*.*

We remark here that the weaker statement that $\mathbf{P}_{\mathbf{P}\text{-comp}}$ has p-measure 0 was proved by Schuler and Yamakami [98], and independently by Cai and Selman [19]. Notice that, as an immediate consequence of Corollary 7.4.6, if $\mathbf{NP} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$, then $\mathbf{NP}$ has p-measure 0.

Definition 7.4.7 (Almost Immunity) [24, 77] A set is called *almost* $\mathbf{P}$*-immune* if it is a union of a

P set and a **P**-immune set. A set whose complement is almost **P**-immune is called **P**-*levelable*.

For a set A, a set S is called a maximal subset of A if any infinite subset of A in **P** is a finite variant of S, i.e., $\|S - A\|$ is finite. It is known that, for any infinite recursive set A, A is almost **P**-immune if and only if A has a maximal subset in **P** [24, 77].

Lemma 7.4.8 **[24, 77]** *If a set A not in **P** satisfies $A \leq_m^{\mathrm{p}} A$ via a length-increasing reduction, then A is **P**-levelable. Hence, most well-known **NP**-complete sets are **P**-levelable unless **P** = **NP**.*

Proof. To arrive at a contradiction, we assume that A is almost **P**-immune. The almost-**P**-immunity ensures that there is a maximal subset of A in **P**. Let E be such a set.

Now we define $B = \{x \mid x \notin E \wedge f(x) \in E\}$. Since $f \in \mathbf{FP}$ and $E \in \mathbf{P}$, we conclude that B is in **P**. Moreover, $B \cap E = \emptyset$ by definition. We now show that B is infinite. Assume that B is finite. We choose an element x in $A - E$ such that $B \subseteq \Sigma^{|x|-1}$. The element x exists because $A \notin \mathbf{P}$ but $E \in \mathbf{P}$. Consider the set $E_x = \{x\} \cup \{f^{(k)}(x) \mid k \geq 1\}$, where $f^{(1)}(x) = f(x)$ and $f^{(k+1)}(x) = f(f^{(k)}(x))$ for each $k > 0$. Since f reduces A to A, $E_x \subseteq A$. Moreover, E_x is infinite and in **P** since f is length-increasing (i.e., $|f(x)| > |x|$). Note that $E_x \cap E \neq \emptyset$ since, otherwise, $E \cup E_x$ is a **P**-subset of A such that $E - E_x$ is infinite, and consequently E is not a maximal subset of A. Since $x \notin E$, there exists a string y in $E_x - E$, but $f(y) \in E$. Hence, $y \in B$. Clearly $|y| > |x|$. This contradicts the finiteness of B.

The latter part of the claim follows from the fact that most known **NP**-complete sets A, such as SAT, satisfy the condition $A \leq_m^{\mathrm{p}} A$ via a length-increasing reduction. □

Proposition 7.4.9 *Assume that **P** ≠ **NP**. If every set in $\mathbf{P}_{\mathbf{P}\text{-comp}} - \mathbf{P}$ is almost **P**-immune, then **NP** $\not\subseteq$ $\mathbf{P}_{\mathbf{P}\text{-comp}}$.*

Proof. Consider the **NP**-complete set SAT. If **P** ≠ **NP**, then SAT is in **NP** − **P** and is **P**-levelable by Lemma 7.4.8. Hence, if SAT $\in \mathbf{P}_{\mathbf{P}\text{-comp}}$, then $\mathbf{P}_{\mathbf{P}\text{-comp}} - \mathbf{P}$ contains a set which is not almost **P**-immune. □

To show that all sets in $\mathbf{P}_{\mathbf{P}\text{-comp}} - \mathbf{P}$ are almost **P**-immune seems difficult. Moreover, we do not know how to construct a **P**-levelable set in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ which is not in **P**.

7.5 Closure Properties

This section will be devoted to closure properties under several types of polynomial-time reductions. Recall that, for a reducibility $\leq_\alpha$, a class $\mathcal{C}$ is *closed (downward) under $\leq_\alpha$-reductions* if, for every two sets A and B, $A \leq_\alpha B$ and $B \in \mathcal{C}$ implies $A \in \mathcal{C}$. We know that most known complexity classes are closed under p-m-reductions, such as **RP**, **BPP**, **NP**, **PH**, **PP**, $\oplus$**P**, etc.

In Subsection 7.5.1, we shall demonstrate that neither $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ nor $\mathbf{\Sigma}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$, $k > 0$, is closed under p-m-reducibility. The lack of this property indicates a structural difference between $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and other regular complexity classes, such as $\mathbf{P}$, $\mathbf{NP}$ and $\mathbf{BPP}$. Thus, it presents a partial solution to the $\mathbf{NP} \subseteq ?\mathbf{P}_{\mathbf{P}\text{-comp}}$ question. Our result relies on the sets constructed in Section 7.3.1.

In Subsection 7.5.2, we shall show that the class $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is not closed under the polynomially-bounded existential operator. This result also implies that $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is different from $\mathbf{\Sigma}_k^{\mathrm{p}}$.

7.5.1 Polynomial Time Reducibilities

In this subsection, we will discuss properties of quintessential complexity classes in relation to various polynomial-time reducibilities.

We claim that there is a the difference between polynomial-time truth-table reducibility and Turing reducibility within $\mathbf{P}_{\mathbf{P}\text{-comp}}$. This claim is proved by demonstrating that the Turing closure of $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is "large," whereas its truth-table closure is "small."

The following proposition shows that the Turing closure of $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is equal to $\mathbf{\Delta}_k^{\mathrm{exp}}$, where $k \geq 0$.

Proposition 7.5.1 **[94]** *For each $k \geq 1$, the Turing closure of $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ is equal to $\mathbf{\Delta}_k^{\mathrm{exp}}$.*

Proof. We shall show that every set in $\mathbf{\Delta}_k^{\mathrm{exp}}$ is p-T-reducible to some set in $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$. Let B be an arbitrary set in $\mathbf{\Delta}_k^{\mathrm{e}}$ and assume that $B \in \mathrm{ATIME}^{\Delta}(k, 2^{cn})$ for some constant $c > 0$. Let A be the set defined in the Hard Instance Lemma. We then define the set B' as

$$B' = \{xy \mid |x| = |y| \wedge xy \in A \wedge y \in B'\}.$$

Since A is in $\mathrm{DTIME}(n^{\log n})$, clearly B' is in $\mathbf{\Delta}_k^{\mathrm{e}}$. Since $B' \subseteq A$, by the Hard Instance Lemma, B' is in $\mathbf{A}\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$. Hence, $B' \in \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$.

From $\|A \cap \Sigma^{2n}\| \leq 1$, it follows that, for all y of length n, $y \in B$ if and only if $h(1^n)y \in B'$. Recall that $h(1^n)$ is computed by n adaptive queries to $Prefix(A)$. This implies that B is p-T-reducible to $B' \oplus Prefix(A)$. Let $C = B' \oplus Prefix(A)$. Since $B' \in \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ and $Prefix(A) \in \mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$, C is also in $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ by Lemma 7.2.12. □

It is not known whether p-T-reducibility in the above proposition can be replaced by p-m-reducibility. Note that this is the case if $\mathbf{\Delta}_k^{\mathrm{p}}{}_{\mathbf{P}\text{-comp}}$ has some p-m-complete sets for $\mathbf{\Delta}_k^{\mathrm{e}}$.

Corollary 7.5.2 **[94]** *The truth-table closure of $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is not equal to $\mathbf{EXP}$.*

Proof. Recall that $\mathbf{EXP}$ has p-measure 1. The corollary then is immediate from Corollary 7.4.6 since, otherwise, $\mathbf{EXP}$ has p-measure 0, a contradiction. □

Assume that there exists a p-m-reduction from a set A to another set B which is in $\mathbf{P}_{\mathbf{P}\text{-comp}}$. The next proposition characterizes the reduction when $A \notin \mathbf{P}$.

Proposition 7.5.3 *Assume that a set A is p-m-reducible to B in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ via a reduction function f. If A is not in $\mathbf{P}$, then, for every number c, there are infinitely many strings x satisfying the inequality $\log|x| > c|f(x)|$.*

Proof. Assume that A is p-m-reducible to B in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ via a $\mathbf{P}$-computable reduction function f. Furthermore, assume that there exists a positive constant c satisfying $\log|x| \leq c|f(x)|$ for almost all x. In the following, we want to show that A belongs to $\mathbf{P}$.

Let M_B be a machine which computes B in polynomial time on ν_{stand}-average, and consider the following machine M_A: on input x, compute $f(x)$, and accept the input exactly when M_B accepts $f(x)$. We thus have:

$$\text{Time}_{M_A}(x) \leq d \cdot (\text{Time}_{M_f}(x) + \text{Time}_{M_B}(f(x)) + 1)$$

for some positive constant d. Since $B \in \mathbf{P}_{\mathbf{P}\text{-comp}}$, $\text{Time}_{M_B}(f(x)) \leq 2^{c'|f(x)|}$ for some constant $c' > 0$. This implies that

$$\text{Time}_{M_B}(f(x)) \leq 2^{c'|f(x)|} \leq |x|^{c'/c}$$

for almost all x, since $2^{|f(x)|} \geq |x|^{1/c}$. This shows that A belongs to $\mathbf{P}$. □

The next proposition, due to Wang and Belanger [6] (in a different setting) and Schuler and Yamakami [98], shows the closure property of the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ under restricted reductions.

Proposition 7.5.4 **[6, 98]** *For $k > 0$, the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is closed under increasing hp-m-reductions.*

Proof. Let B be any set in $\mathbf{\Delta}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$. For a set A, we assume that $A \leq^{\text{hp}}_{m} B$ via a $\mathbf{P}$-computable reduction f which is increasing and p-honest. We shall show that A is also in $\mathbf{\Delta}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$. For each $\mu \in \mathbf{P}$-comp, let $\nu = \mu_{f^{-1}}$. By the condition on f, we conclude that $\nu \in \mathbf{P}$-comp by Lemma 4.2.8. It is easy to see that if B is computed by a deterministic Turing machine M in polynomial time on ν-average, then A is computable in polynomial time on μ-average by simulating $M(f(x))$. □

As an example application of this proposition, consider the bounded halting problem BHP. Note that BHP is p-1-complete for $\mathbf{NP}$ under increasing, p-honest reductions. Therefore, by Proposition 7.5.4, if BHP $\in \mathbf{P}_{\mathbf{P}\text{-comp}}$, then $\mathbf{NP}$ is included in $\mathbf{P}_{\mathbf{P}\text{-comp}}$.

Next we show that neither $\mathbf{\Delta}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$ nor $\mathbf{\Sigma}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$ is closed under p-m-reducibility. Note that Wang and Belanger [111] have shown a similar result for their classes AP_{P}, ANP_{P} and DNP of distributional decision problems.

Theorem 7.5.5 *Let $k > 0$. Neither $\mathbf{\Delta}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$ nor $\mathbf{\Sigma}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$ is closed downward under p-m-reducibility.*

Proof. We prove the theorem only for $\mathbf{\Delta}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$, since the proof for $\mathbf{\Sigma}^{\text{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$ is almost identical. Let A be the set in $\text{DTIME}(O(n^{\log^* n}))$ such that $\|A^n\| = 1$ for all $n \in \mathbb{N}$ and let h be the function h computable in

time $O(n^{\log^* n})$ in order to separate $\mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp}$ from $\mathrm{ATIME}^{\Delta}(k, O(2^n))$, both of which are used in Theorem 7.3.2.

Now we define a reduction function f as follows:

$$f(x) = \begin{cases} h(1^{\lfloor \log |x| \rfloor^2}) & \text{if } x \in \{0^{2^n} \mid n \in \mathbb{N}\}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $f(0^{2^n}) = x$, then $n \leq \sqrt{|x|}$. Let $T = \{0^{2^i} \mid i \in \mathbb{N}, f(0^{2^i}) \in A\}$. Clearly $T \leq_m^{\mathrm{p}} A$ via f. Note that the function f is not p-honest.

We shall show that f is computable in time $O(n)$. Assume that $x = 0^{2^i}$ for some $i \in \mathbb{N}$, and let y be a string of length $\lfloor \log |x| \rfloor^2$ in A. By our definition, the computation time of $f(x)$, where $|x| = n$, is at most

$$c \cdot |y|^{\log^* |y|} \leq c \cdot 2^{\log^2 |y|} \leq c \cdot 2^{(\log \log^2 |x|)^2} \leq c \cdot 2^{4(\log \log n)^2} \leq n$$

for any sufficiently large n.

We next show that $T \notin \mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp}$. Assume otherwise. Since T is tally, T is in $\mathbf{\Delta}_k^{\mathrm{p}}$. Note that $\mathbf{\Delta}_k^{\mathrm{p}} = \mathrm{ATIME}^{\Delta}(k, n^{O(1)})$. Consider the following algorithm:

begin
  input x
  if $x \notin A$ **then** reject
  take n $(\leq \sqrt{|x|})$ such that $f(0^{2^n}) = x$
  if $0^{2^n} \in T$ **then** accept **else** reject
end.

This algorithm computes A and is in $\mathrm{ATIME}^{\Delta}(k, O(2^n))$. This clearly gives a contradiction. Therefore, $T \notin \mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp}$. □

Theorem 7.5.5 yields the significant consequence that the classes in the real polynomial-time hierarchy under **P**-comp are *structurally* different from most known worst-case complexity classes.

Corollary 7.5.6 *Let $k > 0$ and let $\mathcal{C}$ be any complexity class. If $\mathcal{C}$ is closed downward under p-m-reductions, then $\mathbf{\Delta}_k^{\mathrm{p}}\mathbf{P}\text{-comp} \neq \mathcal{C}$ and $\mathbf{\Sigma}_k^{\mathrm{p}}\mathbf{P}\text{-comp} \neq \mathcal{C}$.*

As another application of Theorem 7.5.5, we shall present the following corollary, proven as Lemma 7.1 in [36]. The corollary demonstrates the necessity of the p-honesty condition in Condition I′ defined in Section 4.7. The proof presented here is very different from Gurevich's and is based on Theorem 7.5.5.

Corollary 7.5.7 **[36]** *There exist a function $f \in \mathbf{FP}$ and a distribution $\mu \in \mathbf{P}$-comp such that, for every $\nu \in \mathbf{P}$-comp and every function p which is polynomial on μ-average, $\hat{\nu}(y) < \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for some y.*

Proof. Assuming the contrary of the corollary, we shall prove that $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is closed under p-m-reductions. This contradicts Theorem 7.5.5.

Assume that A is p-m-reducible to B via a reduction f, and also B is in $\mathbf{P}_{\mathbf{P}\text{-comp}}$. We show that $A \in \mathbf{P}_{\mathbf{P}\text{-comp}}$. For every distribution μ in $\mathbf{P}$-comp, by our assumption, there is a distribution $\nu \in \mathbf{P}$-comp and a function p which is polynomial on μ-average such that $\hat{\nu}(y) \geq \sum_{x \in f^{-1}(y)} \frac{\hat{\mu}(x)}{p(x)}$ for all y. Note that B is computable in polynomial time on ν-average. Then A is also computable in polynomial time on μ-average. Since μ is arbitrary, A belongs to $\mathbf{P}_{\mathbf{P}\text{-comp}}$. □

Let us recall Condition I′. Condition I′ asserts that every $\mathbf{P}$-samplable distribution is avp-dominated by some $\mathbf{P}$-computable distribution. Assuming Condition I′, $\mathbf{P}_{\mathbf{P}\text{-comp}}$ turns out to be closed under many-one reducibility with p-honest polynomial-time computable reductions.

Proposition 7.5.8 *If Condition I′ holds, then $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-comp}}$ is closed downward under hp-m-reductions.*

Proof. Immediate from Lemma 6.4.2 and by the same argument in Proposition 7.5.7. □

Finally we show the existence of an incomparable pair in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ with respect to the hp-m-reducibility.

Lemma 7.5.9 **[98]** *There exists a pair of sets A and B in $\mathbf{P}_{\mathbf{P}\text{-comp}}$ such that $A \not\leq^{\mathrm{hp}}_m B$ and $B \not\leq^{\mathrm{hp}}_m A$.*

Proof. Recall from Theorem 7.4.1 that there is a $\mathbf{P}$-immune set in $\mathbf{P}_{\mathbf{P}\text{-comp}}$. Let A be such a set, and let $B = \{0\}^*$. Note that $A \notin \mathbf{P}$. If $A \leq^{\mathrm{hp}}_m B$, then $A \in \mathbf{P}$ since $B \in \mathbf{P}$, a contradiction. If $B \leq^{\mathrm{hp}}_m A$ via a polynomially honest $f \in \mathbf{FP}$, then A should not be $\mathbf{P}$-immune since the infinite set $\{f(x) \mid x \in B\}$, which is a subset of A, belongs to $\mathbf{P}$. Both cases induce contradictions. □

Let us next observe the closure property of average complexity classes under hp-m-reductions when $\mathbf{P}$-samplable distributions are taken instead. The $\mathbf{P}$-samplable distributions show us a different world.

Proposition 7.5.10 *For any $k > 0$, $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-samp}}$ and $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{IP}\text{-samp}}$ are both closed under hp-m-reductions. A similar result holds for $\mathbf{\Sigma}^{\mathrm{p}}_{k\,\mathbf{P}\text{-samp}}$ and $\mathbf{\Sigma}^{\mathrm{p}}_{k\,\mathbf{IP}\text{-samp}}$.*

Proof. We only consider the case $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-samp}}$. Assume that A is p-m-reducible to B via a p-honest reduction f, and B is in $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-samp}}$. We shall show that A is in $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-samp}}$.

Let μ be any distribution in $\mathbf{P}$-samp and define ν as $\nu = \mu_{f^{-1}}$. Distribution ν may not be $\mathbf{P}$-samplable, but by Corollary 4.4.12, we can find another distribution ν' which p-dominates ν and is in $\mathbf{P}$-samp. Since $B \in \mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-samp}}$, we have $(B, \nu') \in \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, *)$. Note that $(A, \mu) \leq^{\mathrm{p}}_m (B, \nu')$ via f. Hence, by Lemma 6.4.2(2), we have $(A, \mu) \in \mathrm{Aver}(\mathbf{\Delta}^{\mathrm{p}}_k, *)$. Hence, A is in $\mathbf{\Delta}^{\mathrm{p}}_{k\,\mathbf{P}\text{-samp}}$. □

7.5.2 Polynomially Bounded Existential Operator

We have already shown the difference between $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and other regular complexity classes, such as **NP** and **BPP**, by demonstrating that $\mathbf{P}_{\mathbf{P}\text{-comp}}$ is not closed under p-m-reductions. Here we shall discuss other properties that elucidate the structural difference between $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and **NP** as well as **UP** and **PP**.

We shall introduce the following types of existential operators and show that $\mathbf{\Delta}^{\mathrm{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$ is not closed under such operators.

Definition 7.5.11 (Polynomially Bounded Operators) Let C be a complexity class.

1. A set S is in $\exists^{\mathrm{p}}\cdot\mathcal{C}$, the *closure of* $\mathcal{C}$ *under the existential operator*, if there exist a polynomial p and a set $B \in \mathcal{C}$ such that $S = \{x \mid \exists y \in \Sigma^{p(|x|)}[\langle y, x\rangle \in B]\}$.

2. A set S is in $\mathrm{U}^{\mathrm{p}}\cdot\mathcal{C}$, the *closure of* $\mathcal{C}$ *under the unique existential operator*, if there exist a polynomial p and a set $B \in \mathcal{C}$ such that if $x \in S$ then $\exists! y \in \Sigma^{p(|x|)}[\langle y, x\rangle \in B]$; otherwise $\forall y \in \Sigma^{p(|x|)}[\langle y, x\rangle \notin B]$.

Obviously $\mathrm{U}^{\mathrm{p}}\cdot\mathcal{C} \subseteq \exists^{\mathrm{p}}\cdot\mathcal{C}$ for any complexity class $\mathcal{C}$. It also holds that $\exists^{\mathrm{p}}\cdot\mathbf{\Sigma}^{\mathrm{p}}_{k} \subseteq \mathbf{\Sigma}^{\mathrm{p}}_{k}$ and $\mathrm{U}^{\mathrm{p}}\cdot\mathbf{UP} \subseteq \mathbf{UP}$. We need the following lemma for subsequent results.

Lemma 7.5.12 *Let $\mathcal{C} \in \{\mathbf{\Delta}^{\mathrm{p}}_{k}, \mathbf{\Sigma}^{\mathrm{p}}_{k}, \mathbf{\Pi}^{\mathrm{p}}_{k} \mid k > 0\}$ and let g be any increasing function on Σ^* which belongs to* **FP**. *If $A \in \mathcal{C}_{\mathbf{P}\text{-comp}}$, then the set $B = \{\langle w, x\rangle \mid w = g(x) \wedge x \in A\}$ is also in $\mathcal{C}_{\mathbf{P}\text{-comp}}$.*

Proof. Here we prove the lemma for the case $\mathcal{C} = \mathbf{\Delta}^{\mathrm{p}}_{k}$. Assume that $A \in \mathbf{\Delta}^{\mathrm{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$ and $B \notin \mathbf{\Delta}^{\mathrm{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$. There exist a polynomial q and a deterministic Turing machine M_g computing g in time q. Since $B \notin \mathbf{\Delta}^{\mathrm{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$, there exists a distribution $\mu \in \mathbf{P}$-comp such that, for any machine N and any polynomial s, $\hat{\mu}(\{x \mid \mathrm{Time}_N(x) > s(|x| \cdot r)\}) \geq 1/r$ for some real number $r \geq 1$. Recall the monotonicity of the pairing function $\langle , \rangle$ for any increasing function (see Section 2.5). For this function g, $x < y$ implies $\langle g(x), x\rangle < \langle g(y), y\rangle$. Moreover, $\langle g(x^-), x^-\rangle \leq \langle g(x), x\rangle^-$.

Define ν as $\nu(x) = \mu(g(x), x)$. This function ν is indeed a distribution in $\mathbf{P}$-comp since $\nu(x) \geq \nu(x^-)$. Moreover, we have $\hat{\nu}(x) \geq \hat{\mu}(g(x), x)$. This is seen as follows:

$$\begin{aligned}\hat{\nu}(x) &= \nu(x) - \nu(x^-) = \mu(\langle g(x), x\rangle) - \mu(\langle g(x^-), x^-\rangle) \\ &\geq \mu(\langle g(x), x\rangle) - \mu(\langle g(x), x\rangle^-) = \hat{\mu}(g(x), x).\end{aligned}$$

Since $A \in \mathbf{\Delta}^{\mathrm{p}}_{k}\mathbf{P}_{\mathbf{P}\text{-comp}}$, there exist a semi-deterministic alternating Turing machine M computing A with k-alternation and a polynomial p such that $\hat{\nu}(\{x \mid \mathrm{Time}_M(x) > p(|x| \cdot r)\}) < 1/r$ for all $r > 0$. Assume that $p(n) > n$ for all n.

Now consider a machine N which computes B in the following fashion: on input $\langle w, x\rangle$, accept the input if $w = g(x)$ and M accepts x, and reject the input otherwise. Note that, by our assumption, N does not halt in polynomial time on μ-average. To arrive at a contradiction, however, we show that N runs in polynomial time on μ-average. Note that $\mathrm{Time}_N(\langle w, x\rangle) \leq c(\mathrm{Time}_M(x) + \mathrm{Time}_{M_g}(x) + |x| + 1)$ for some constant $c > 0$;

on the other hand, $\mathrm{Time}_N(\langle w,x\rangle) \le c(p(|x|)+q(|x|)+1)$ for all w different from $0^{p(|x|)}$. Let s be a polynomial such that

$$s(|\langle w,x\rangle| \cdot r) \ge c(p(|x| \cdot r) + q(|x|) + |x| + 1).$$

For all $r \ge 1$, we have

$$\begin{aligned}
&\hat{\mu}(\{y \mid \mathrm{Time}_N(y) > s(|y| \cdot r)\}) \\
&\quad\le \hat{\mu}(\{\langle w,x\rangle \mid w \neq g(x) \wedge \mathrm{Time}_N(\langle w,x\rangle) > s(|\langle w,x\rangle| \cdot r)\}) \\
&\qquad\quad +\hat{\mu}(\{\langle w,x\rangle \mid \mathrm{Time}_N(\langle w,x\rangle) > s(|\langle w,x\rangle| \cdot r)\}) \\
&\quad\le \hat{\mu}(\{\langle w,x\rangle \mid \mathrm{Time}_M(x) > p(|x| \cdot r)\}) \\
&\quad\le \hat{\nu}(\{x \mid \mathrm{Time}_M(x) > p(|x| \cdot r)\}) \quad < \quad \frac{1}{r}.
\end{aligned}$$

Hence, N is polynomial-time bounded on μ-average. This contradicts our assumption. □

Contrary to the situation for $\mathbf{\Delta}^{\mathrm{p}}_k$, it is not known whether $\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} \subseteq \mathrm{U}^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$, or even whether $\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} \subseteq \exists^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$. However, we are able to show that the converse does not hold for $\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$.

Theorem 7.5.13 *Let $k > 0$. $\mathrm{U}^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} \not\subseteq \mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$. Hence, $\exists^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} \not\subseteq \mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$.*

Proof. Assume that $\mathrm{U}^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} \subseteq \mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$. By Theorem 7.3.2(1), we can define a set A in $\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} - \mathrm{ATIME}^{\Delta}(k, 2^n)$ and take a corresponding function h, as defined in the proof of Lemma 7.3.1, which has the following property: for each length n, $A^n \subseteq \{h(1^n)\}$ and $h(1^n)$ is computed in time $O(2^n)$. Now define $C = \{0^n \mid \exists y[|y| = n \wedge y \in A]\}$. To see that $C \in \mathrm{U}^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$, let $B = \{\langle 0^{|x|}, x\rangle \mid x \in A\}$. By Lemma 7.5.12, B is also in $\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$, and let $C = \{0^n \mid \exists y[|y| = n \wedge \langle y,x\rangle \in B]\}$.

By our assumption, the set C is also in $\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$. Since C is tally, by Proposition 7.2.17, C is in $\mathbf{\Delta}^{\mathrm{p}}_k$. As a result, we have $C \in \mathrm{ATIME}^{\Delta}(k, O(2^n))$. Consider the following procedure that computes A:

begin algorithm for A
  input x (say, $|x| = n$)
  if $0^n \notin C$ **then** reject
  compute $h(1^n)$
  if $x = h(1^n)$ **then** accept **else** reject
end.

This procedure guarantees that A is in $\mathrm{ATIME}^{\Delta}(k, O(2^n))$, and hence, this contradicts the choice of A. □

As an immediate consequence of Theorem 7.5.13, we again have $\mathbf{P}_{\mathbf{P}\text{-comp}} \neq \mathbf{NP}$ and $\mathbf{P}_{\mathbf{P}\text{-comp}} \neq \mathbf{UP}$.

We note that it is not known whether $\exists^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} \subseteq \mathbf{\Sigma}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$. However, if $\exists^{\mathrm{p}}\cdot\mathbf{\Delta}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}} \subseteq \mathbf{\Sigma}^{\mathrm{p}}_{k\mathbf{P}\text{-comp}}$, then Theorem 7.5.13 implies that $\mathbf{P}_{\mathbf{P}\text{-comp}} \neq \mathbf{NP}_{\mathbf{P}\text{-comp}}$, and thus that $\mathbf{P} \neq \mathbf{NP}$.

Corollary 7.5.14 *$\exists^{\mathrm{p}}\cdot\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{NP}_{\mathbf{P}\text{-comp}}$ if $\mathbf{P} = \mathbf{NP}$.*

Moreover, we can consider a restricted type of probabilistic operator.

Definition 7.5.15 (Class $\mathrm{P}^{\mathrm{p}\cdot}\mathcal{C}$) For a complexity class $\mathcal{C}$, a set S is in $\mathrm{P}^{\mathrm{p}\cdot}\mathcal{C}$, the *closure under the probabilistic operator*, if there are a polynomial p and a set $B \in \mathcal{C}$ such that $S = \{x \mid \|\{y \in \Sigma^{p(|x|)} \mid \langle y, x\rangle \in B\}\| > \frac{1}{2} \cdot 2^{p(|x|)}\}$.

The probabilistic class **PP** is closed under this operator, i.e., $\mathrm{P}^{\mathrm{p}\cdot}\mathbf{PP} \subseteq \mathbf{PP}$.

A proof technique similar to that used for Theorem 7.5.13 shows the following theorem.

Theorem 7.5.16 **[98]** $\mathrm{P}^{\mathrm{p}\cdot}\mathbf{P}_{\mathbf{P}\text{-comp}} \not\subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$.

Proof. Let the set A and the function h be as defined in the proof of Theorem 7.5.13 but with $k = 1$. Recall the algorithm described in the proof of Lemma 7.3.1. Define B to be a set accepted by the following algorithm:

begin
  input x (say, $|x| = n$)
  compute $z_1^n = \min\{w \in \Sigma^{l(n)} \mid w \notin \mathrm{KT}[n, 2^{s(i)\cdot n}]\}$
  if $1 \sqsubseteq z_1^n$ **then** $y = 0$ **else** $y = 1$
  if $x \in y\Sigma^{|x|-1}$ **then** accept **else** reject
end.

By the choice of y in the algorithm, $A \cap B = \emptyset$ and $\|B \cap \Sigma^n\| = \frac{1}{2} \cdot 2^n$ for all n. Note that B is in **P**. Now we let $A' = A \cup B$. Then, we have $A' \in \mathbf{P}_{\mathbf{P}\text{-comp}}$. Note that either $\|A' \cap \Sigma^n\| > \frac{1}{2} \cdot 2^n$ or $\|A' \cap \Sigma^n\| = \frac{1}{2} \cdot 2^n$.

Let $C = \{0^n \mid \|\{y \in \Sigma^n \mid y \in A'\}\| > \frac{1}{2} \cdot 2^n\}$. We have $C \in \mathrm{P}^{\mathrm{p}\cdot}\mathbf{P}_{\mathbf{P}\text{-comp}}$ by definition, but $C \notin \mathbf{P}_{\mathbf{P}\text{-comp}}$ since if $C \in \mathbf{P}_{\mathbf{P}\text{-comp}}$, then the same algorithm as in the proof of Theorem 7.5.13 computes A in time $O(2^n)$. □

Theorem 7.5.16 directly shows that $\mathbf{P}_{\mathbf{P}\text{-comp}} \neq \mathbf{PP}$.

7.6 Bounded Error Probabilistic Polynomial Time

We direct our attention now to the bounded-error probabilistic complexity class $\mathbf{BPP}_{\mathcal{F}}$. In this section, we shall discuss the class $\mathbf{BPP}_{\mathcal{F}}$ under weak reductions, and consider a result of Schuler and Watanabe [96] regarding the question $\mathbf{NP} \subseteq ?\mathbf{BPP}_{\mathcal{F}}$.

We begin with the closure property of $\mathbf{BPP}_{\mathbf{P}\text{-samp}}$ under p-honest bpp-tt-reductions.

Proposition 7.6.1 $\mathbf{BPP}_{\mathbf{P}\text{-samp}}$ *is closed under p-honest, bpp-tt-reducibility.*

Proof. Assume that $A \in \mathbf{BPP}_{\mathbf{P}\text{-samp}}$ and A is bpp-tt-reducible to B via a p-honest reduction. Let

M be such a reduction, i.e., a polynomial-time probabilistic Turing machine computing A with bounded error which accesses oracle B with nonadaptive p-honest queries. For simplicity, assume that the number of queries on each computation path is of the form 2^m, where m is dependent only on x. Since M makes p-honest queries, for any query z by M on x, $|x|$ is bounded above by an absolute polynomial p in $|z|$.

Choose any distribution μ in **P**-samp. We define a distribution ν as follows:

$$\hat{\nu}(z) = \sum_{x:|x|\leq p(|z|)} \hat{\mu}(x) \cdot \mathbf{Pr}_M[M \text{ on } x \text{ queries } z\ ].$$

It is not difficult to confirm that ν is **P**-samplable. To see this, let M_μ be a sampling machine which samples μ and consider the following sampling algorithm:

begin sampling algorithm
  input 0^i
  simulate M_μ on 0^i
  let x be an output of M_μ on 0^i
  list all query strings, say $z_1, z_2, \ldots, z_{2^m}$
  generate k $(1 \leq k \leq 2^m)$ at random
  output z_k
end.

Hence, $(A, \nu) \in \mathrm{Aver}(\mathbf{BPP}, \mathbf{P}\text{-samp})$. We now show that $(B, \mu) \leq_{tt}^{\mathrm{bpp}} (A, \nu)$ via M. It is sufficient to check the domination condition for M. This is easily verified, however. By Proposition 5.5.7(4), (B, μ) belongs to $\mathrm{Aver}(\mathbf{BPP}, \mathbf{P}\text{-samp})$. □

Ben-David *et al.* [9] showed that all distributional **NP** search problems are "randomly" reducible to their corresponding distributional decision problems. Later Schuler and Watanabe [96] rephrased this result in the following fashion.

Corollary 7.6.2 **[9, 96]** $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-samp}}$ *if and only if* $\mathbf{\Theta}_2^{\mathrm{p}} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-samp}}$.

Proof. Assume that $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-samp}}$. For any set $A \in \mathbf{\Theta}_2^{\mathrm{p}}$, there exists a set $B \in \mathbf{NP}$ such that A is p-tt-reducible to B. This reduction can be p-honest by choosing an appropriate set B. Hence, A is bpp-tt-reducible to B via some p-honest reduction. Notice that B belongs to $\mathbf{BPP}_{\mathbf{P}\text{-samp}}$ by our assumption. By Proposition 7.6.1, $B \in \mathbf{BPP}_{\mathbf{P}\text{-samp}}$ implies $A \in \mathbf{BPP}_{\mathbf{P}\text{-samp}}$. Therefore, we get $A \in \mathbf{BPP}_{\mathbf{P}\text{-samp}}$. □

Impagliazzo and Levin [44] demonstrated that distributional **NP** decision problems are "reducible" to distributional **NP** decision problems with standard distribution. Schuler and Watanabe [96] extended their result in terms of quintessential computability.

Theorem 7.6.3 **[96]** *Given a subset* $\mathcal{F}$ *of* **avP**-samp *including* ν_{stand}, $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathcal{F}}$ *if and only if* $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{avP}\text{-samp}}$.

Proof. We follow the argument given by Schuler and Watanabe [96]. Let us assume that $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{avP}\text{-samp}}$.

Let (A,μ) be an arbitrary distributional decision problem from $\mathrm{Dist}(\mathbf{NP}, \mathbf{avP}\text{-samp})$. We shall show that (A,μ) is in $\mathrm{Aver}(\mathbf{BPP}, *)$. If $\hat{\mu}(\Sigma^n) > 1/2$ for some $n \in \mathbb{N}$, then we can eliminate all strings of length n from the following argument. Hence, we assume without loss of generality that $\hat{\mu}(\Sigma^n) \leq 1/2$ for all $n \in \mathbb{N}$.

Since $A \in \mathbf{NP}$, there are a set $E \in \mathbf{P}$ and a polynomial p such that $A = \{x \mid \exists y \in \Sigma^{p(|x|)}[\langle x,y\rangle \in E]\}$. Moreover, from the fact that $A \in \mathbf{EXP}$, we can define a deterministic Turing machine M_A which recognizes D in time $2^{q(|x|)}$, where x is any input. For the sake of convenience, we assume that $q(n) \geq 32n^3(n+1)^2$ for all $n \in \mathbb{N}$.

Since μ is an average $\mathbf{P}$-samplable distribution, there are a generator M for μ and a polynomial p' such that (i) $|\hat{\mu}(x) - \mathbf{Pr}_M[M(0^i) = x]| \leq 2^{-i}$, and (ii) for every $r > 0$ and $n \in \mathbb{N}$,

$$2^{-2\mathrm{llog}(n)-1} \cdot \mathbf{Pr}_s[M(0^i;s) \in \Sigma^n \wedge \mathrm{Time}_M(0^i;s) > p'(r\cdot(n+i)) \mid s \in \Omega_M(0^i)] < 1/r.$$

Write $p_j(n)$ for $p'(2^{j+1}(n+q(n)))$. Let us define the function f as follows: for each j with $0 \leq j < q(n)$,

$$f_n^{(j)}(s) = \begin{cases} M(0^{q(n)+1};s) & \text{if } \mathrm{Time}_M(0^{q(n)+1};s) \leq p_j(n),\ |M(0^{q(n)+1};s)| = n, \text{ and } j = 0, \\ M(0^{q(n)+1};s) & \text{if } p_{j-1}(n) < \mathrm{Time}_M(0^{q(n)+1};s) \leq p_j(n), \\ & \qquad |M(0^{q(n)+1};s)| = n, \text{ and } 0 < j < q(n), \\ \lambda & \text{otherwise.} \end{cases}$$

Without loss of generality, we may assume that, for any two random inputs s and s', if $|f_n^{(j)}(s)| = |f_n^{(j)}(s')|$, then $|s| = |s'|$, and thus, $\|\{s \mid f_n^{(j)}(s) \in \Sigma^n\}\| < 2^{p_j(n)-1}$.

Fix any integer j satisfying $0 \leq j < q(n)$ and any string x of length n. Let $k_x^{(j)} = \mathrm{ilog}(\|(f_n^{(j)})^{-1}(x)\|)$. Since $\|(f_n^{(j)})^{-1}(x)\| < 2^{p_j(n)-1}$, it follows that $0 \leq k_x^{(j)} < p_j(n)$. As a consequence, we have

$$\begin{aligned} \mathbf{Pr}_M[M(0^{q(n)+1}) = x] &\leq \sum_{j=0}^{q(n)-1} \frac{\|(f_n^{(j)})^{-1}(x)\|}{2^{p_j(n)}} \\ &\qquad + \mathbf{Pr}_s[\mathrm{Time}_M(0^{q(n)+1};s) > p_{q(n)-1}(n) \mid s \in \Omega_M(0^{q(n)+1})] \\ &\leq \sum_{j=0}^{q(n)-1} 2^{-p_j(n)+k_x^{(j)}} + 2^{-2\mathrm{llog}(n)-1} \cdot 2^{-q(n)} \\ &\leq \sum_{j=0}^{q(n)-1} 2^{-p_j(n)+k_x^{(j)}} + 2^{-q(n)-1}. \end{aligned}$$

Therefore, we obtain

$$\hat{\mu}(x) \leq 2^{-q(n)-1} + \mathbf{Pr}_M[M(0^{q(n)+1}) = x] \leq 2^{-q(n)} + \sum_{0\leq j<q(n)} 2^{-p_j(n)+k_x^{(j)}}.$$

For each x, since $\sum_{0\leq j<q(n)} 2^{-p_j(n)+k_x^{(j)}} \leq 1$, there exists an index j' such that $\hat{\mu}(x) \leq 2^{-q(n)} + \frac{1}{q(n)} \cdot 2^{-p_{j'}(n)+k_x^{(j')}}$. Let j_x be the minimum of such an index.

Note that if $q(n) \geq p_{j_x}(n) - k_x^{(j_x)} - \mathrm{ilog}(q(n))$, then $2^{-q(n)} \leq 2^{-\mathrm{ilog}(q(n))} \cdot 2^{-p_{j_x}(n)+k_x^{(j_x)}}$; thus,

$$\hat{\mu}(x) \leq 2 \cdot 2^{-2\mathrm{ilog}(q(n))} \cdot 2^{-p_{j_x}(n)+k_x^{(j_x)}} \leq \frac{2}{q(n)} \cdot 2^{-p_{j_x}(n)+k_x^{(j_x)}}.$$

On the other hand, if $q(n) < p_{j_x}(n) - k_x^{(j_x)} - \mathrm{ilog}(q(n))$, then $\hat{\mu}(x) \leq 2 \cdot 2^{-q(n)} = 2^{-q(n)+1}$.

Let $X_n^{(j)} = \{x \in \Sigma^n \mid j = j_x \wedge q(n) \geq p_{j_x}(n) - k_x^{(j_x)} - \mathrm{ilog}(q(n))\}$. It follows that

$$\begin{aligned}
\hat{\mu}(X_n^{(j)}) &\leq \frac{2}{q(n)} \cdot \sum_{x \in X_n^{(j)}} 2^{-p_j(n)+k_x^{(j)}} = \frac{2}{q(n)} \cdot \sum_{x \in X_n^{(j)}} \frac{\|(f_n^{(j)})^{-1}(x)\|}{2^{p_j(n)}} \\
&\leq \frac{2}{q(n)} \cdot \mathbf{Pr}_s[f_n^{(j)}(s) \in X_n^{(j)} \mid s \in \Omega_M(0^{q(n)+1})] \\
&\leq \frac{2}{q(n)} \cdot \mathbf{Pr}_s[\mathrm{Time}_M(0^{q(n)+1}; s) > p_{j-1}(n) \wedge f_n^{(j)}(s) \neq \lambda \mid s \in \Omega_M(0^{q(n)+1})], \\
&< \frac{1}{n^3} \cdot 2^{-j}
\end{aligned}$$

since $q(n) \geq 2n^3$. Therefore, $\hat{\mu}(X_n^{(j)}) < \frac{1}{n^3} \cdot 2^{-j}$ for all j with $0 \leq j < q(n)$.

Recall that s_n denotes the nth string of Σ^* (N.B. λ is the 0th string), and s_k^m represents the kth string of $\Sigma^{\mathrm{ilog}(m)}$ (N.B. $s_0^m = 0^{\mathrm{ilog}(m)}$). Note that $|s_n| = \mathrm{llog}(n)$.

We begin with the following **NP** sets:

$$B_1 = \{\langle s_1, s_n, s_0^n, s_k^m, h_1 h_2 z\rangle \mid \exists v \in \Sigma^l [h_2 \circ f_n^{(j)} \circ h_1(v) = z] \wedge h_1 \in H_{l,m} \wedge h_2 \in H_{n,l+2}\};$$

$$B_2 = \{\langle s_2, s_n, s_0^n, s_k^m, h_1 h_2 z\rangle \mid \exists v, w \in \Sigma^l [f_n^{(j)} \circ h_1(v) \neq f_n^{(j)} \circ h_1(w) \wedge h_2 \circ f_n^{(j)} \circ h_1(v) = h_2 \circ f_n^{(j)} \circ h_1(w) = z]\};$$

$$B_3 = \{\langle s_3, s_n, s_i^n, s_k^m, h_1 h_2 z\rangle \mid \exists v \in \Sigma^l [z = h_2 \circ f_n^{(j)} \circ h_1(v) \wedge \text{ the } i\text{th bit of } f_n^{(j)} \circ h_1(v) \text{ is 1 }]\}; \text{ and}$$

$$B_4 = \{\langle s_3, s_n, s_i^n, s_k^m, h_1 h_2 z\rangle \mid \exists w \exists v \in \Sigma^l [z = h_2 \circ f_n^{(j)} \circ h_1(v) \wedge \langle f_n^{(j)} \circ h_1(v), w\rangle \in E]\},$$

where $l = m - k$, $h_1 \in H_{l,m}$, and $h_2 \in H_{n,l+2}$. The notation $h_1 h_2 z$ above means the concatenation of three strings h_1, h_2, and z. Let $B = \bigcup_{i=1}^4 B_i$. Clearly B is in **NP**. By the assumption, $(B, \nu_{\mathrm{stand}}) \in \mathrm{Aver}(\mathbf{BPP}, *)$.

Let c_0 be a sufficiently large and fixed constant. First we shall define the randomized Turing machine M_0 that (i) chooses j at random from $\{0, 1, \ldots, q(n) - 1\}$ (by choosing $s_j^{q(n)}$), (ii) chooses k at random from $\{0, 1, \ldots, p_j(n) - 1\}$ (by choosing $s_k^{p_j(n)}$), and (iii) outputs $\langle x, s_j^{q(|x|)} s_k^{p_j(|x|)}\rangle$. The following is a formal description of this algorithm:

begin randomized algorithm for M_0
  input x (say, $n = |x|$)
  compute $\mathrm{ilog}(q(n))$
  generate at random a string s of length $\mathrm{ilog}(q(n))$ such that
    $s = s_j^{q(n)}$ for some j with $0 \leq j < q(n)$
  (assume that $s = s_j^{q(n)}$)
  compute $\mathrm{ilog}(p_j(n))$
  generate at random a string s' of length $\mathrm{ilog}(p_j(n))$ such that

$s' = s_k^{p_j(n)}$ for some k with $0 \leq k < p_j(n)$
(assume that $s' = s_k^{p_j(n)}$)
output $\langle x, ss' \rangle$
end.

The machine M_0 runs in polynomial time because the running time is proportional to the length of the random seeds. The length of each random seed, $|s_j^{q(n)} s_k^{p_j(n)}|$, is at most

$$\mathrm{ilog}(q(n)) + \mathrm{ilog}(p_j(n)) \leq c' \cdot (j + \log n)$$

for some constant $c' > 0$.

Write $d(n, j, k) = s_j^{q(n)} s_k^{p_j(n)}$. Let us define $h(x) = d(n, j_x, k_x^{(j_x)})$ and $D = \{\langle x, h(x) \rangle \mid x \in \Sigma^*\}$. Note that, for any x and y, if $x, y \in X_n^{(j)}$, then $|h(x)| = |h(y)|$. For all $x \in X_n^{(j)}$, since $|h(x)| \leq c'(j + \log n)$, we have $|h(x)| \leq c' \cdot \log(1/n^2 \cdot \hat{\mu}(X_n^{(j)}))$. By Lemma 3.3.11(3), we conclude that $\lambda x.|h(x)|$ is logarithmic on μ-average.

Next we define another randomized oracle Turing machine N as follows:

begin randomized algorithm for N with oracle B
input $\langle x, d(n, j, k) \rangle$ (say, $n = |x|$)
if $n < c_0$ **then** output $A(x)$
set $l = p_j(n) - k$ and $m = p_j(n)$
if $q(n) < l - \mathrm{ilog}(q(n))$ **then** simulate M_A on input x and **halt**
(assume that $q(n) \geq l - \mathrm{ilog}(q(n))$)
choose at random h_1 from $H_{l,p_j(n)}$ and h_2 from $H_{n,l+2}$
let $z = h_2(x)$
check if the following three statements are true:
(i) $\langle s_1, s_n, s_0^n, s_k^m, h_1 h_2 z \rangle \in B$;
(ii) $\langle s_2, s_n, s_0^n, s_k^m, h_1 h_2 z \rangle \notin B$; and
(iii) $x_i = [\langle s_3, s_n, s_i^n, s_k^m, h_1 h_2 z \rangle \in B]$ for all i with $1 \leq i \leq n$
if the three statements as above are not all true
then simulate M_A on input x and **halt**
if $\langle s_4, s_n, s_0^n, s_k^m, h_1 h_2 z \rangle \in B$ **then accept else reject**
end.

It is important to note that our algorithm is error-free as it always outputs a correct answer.

First we consider the running time of the machine N on input from D. Fix a positive integer n not smaller than c_0 and a string x of length n. It suffices to show that $\lambda x.E_s[\mathrm{Time}_N^B(\langle x, h(x) \rangle; s) \mid s \in \Gamma_{N^B}(\langle x, h(x) \rangle)]$ is polynomial on μ-average. Note that $\mathrm{Time}_N^B(\langle x, h(x) \rangle; s) \leq c \cdot (p_j(n) + n + 1)^2$ for some constant $c > 0$ independent of x and s; thus, N is exponential-time bounded in the worst case.

For each j, consider $X_n^{(j)}$. For all $x \in X_n^{(j)}$, the expected running time of the machine N with oracle B

on input $\langle x, h(x)\rangle$ is at most

$$E_s[\mathrm{Time}_N^B(\langle x, h(x)\rangle; s) \mid s \in \Gamma_{N^B}(\langle x, h(x)\rangle)] \leq c \cdot (p_j(n) + n + 1)^2 \leq c' \cdot (2^j n)^d + c'$$

for some appropriate constants $c', d > 0$. Thus, for each $x \in X_n^{(j)}$,

$$E_s[\mathrm{Time}_N^B(\langle x, h(x)\rangle; s) \mid s \in \Gamma_{N^B}(\langle x, h(x)\rangle)] < c' \cdot (2^j n)^d + c' \leq c' \cdot \left(\frac{1}{\tilde{\mu}(X_n^{(j)})n^2}\right)^d + c'.$$

By Lemma 3.3.20, we conclude that $\lambda x.E_s[\mathrm{Time}_N^B(\langle x, h(x)\rangle; s) \mid s \in \Gamma_{N^B}(\langle x, h(x)\rangle)]$ is polynomial on μ-average. Therefore, $\lambda xs.N^B(\langle x, h(x)\rangle; s)$ is polynomial on μ-average.

Next we shall discuss the success probability of machine N. Fix x and let $n = |x|$. Let ρ_x be the probability that the algorithm N on input $\langle x, h(x)\rangle$ succeeds in producing the correct answer. We shall show a lower bound of ρ_x. For this purpose, we introduce a new notion. We say that (h_1, h_2, z) *determines* x if

(i) there exists a string $v \in \Sigma^l$ such that $f_n^{(j)} \circ h_1(v) = x$ and $h_2(x) = z$; and

(ii) for all $w \in \Sigma^l$, if $h_2 \circ f_n^{(j)} \circ h_1(w) = z$, then $f_n^{(j)} \circ h_1(w) = x$.

Fix j and assume that $\|(f_n^{(j)})^{-1}(x)\| \neq 0$. Let us choose $k_x^{(j)}$ and set $l = p_j(n) - k_x^{(j)}$. Then, $\rho_x \geq 1/16$ follows from the claim below.

Claim 18 $\mathbf{Pr}_{h_1 h_2}[(h_1, h_2, h_2(x))$ *determines* $x \mid h_1 \in H_{l,p_j(n)}, h_2 \in H_{n,l+2}] \geq \frac{1}{16}$.

Proof of Claim. Let $\sigma = \mathbf{Pr}_{h_1 h_2}[(h_1, h_2, h_2(x))$ determines $x \mid h_1 \in H_{l,p_j(n)}, h_2 \in H_{n,l+2}]$. To compute σ, we introduce two probabilities:

$$\begin{aligned} \sigma_1 &= \mathbf{Pr}_{h_1}[\exists v \in \Sigma^l[f_n^{(j)} \circ h_1(v) = x]]; \text{ and} \\ \sigma_2(h_1) &= \mathbf{Pr}_{h_2}[\exists v, w \in \Sigma^l[f_n^{(j)} \circ h_1(v) \neq f_n^{(j)} \circ h_1(w) \wedge h_2 \circ f_n^{(j)} \circ h_2(v) = h_2 \circ f_n^{(j)} \circ h_2(w)]]. \end{aligned}$$

By the definition of σ, we have $\sigma \geq \sigma_1 - \max_{h_2 \in H_{n,l+2}} \sigma_2(h_1)$. In the following, we shall estimate these two probabilities σ_1 and $\sigma_2(h_1)$.

We first calculate the probability σ_1. Let us define two sets

$$\begin{aligned} G_x &= \{(s, v) \mid f_n^{(j)}(s) = x \wedge v \in \Sigma^l\}; \text{ and} \\ G'_x &= \{(s, v, s', v') \mid (s, v) \in G_x, (s', v') \in G_x, s \leq s', v \leq v', (s, v) \neq (s', v')\}. \end{aligned}$$

Notice that $\|G_x\| = \|(f_n^{(j)})^{-1}(x)\| \cdot 2^l$ and $\|G'_x\| = \frac{\|G_x\|(\|G_x\|-1)}{2}$. From $l = p_j(n) - k_x^{(j)}$, it follows that

$$2^{p_j(n)-1} \leq 2^{k_x^{(j)}-1} \cdot 2^l \leq \|G_x\| \leq 2^{k_x^{(j)}} \cdot 2^l = 2^{p_j(n)}.$$

The probability σ_1 is estimated as follows:

$$\sigma_1 = \mathbf{Pr}_{h_1}[\exists (s, v) \in G_x[h_1(v) = s] \mid h_1 \in H_{l,p_j(n)}]$$

$$
\begin{aligned}
&\geq \sum_{(s,v)\in G_x} \mathbf{Pr}_{h_1}[h_1(v) = s \mid h_1 \in H_{l,p_j(n)}] \\
&\qquad - \sum_{(s,v,s',v')\in G'_x} \mathbf{Pr}_{h_1}[h_1(v) = s \wedge h_1(v') = s' \mid h_1 \in H_{l,p_j(n)}] \\
&= \|G_x\| \cdot 2^{-p_j(n)} - \|G'_x\| \cdot 2^{-2p_j(n)} \\
&\geq 2^{-2p_j(n)} \cdot \left(\|G_x\|^2 - \frac{\|G_x\|(\|G_x\|-1)}{2} \right) \\
&\geq 2^{-2p_j(n)} \cdot \frac{\|G_x\|^2}{2} \;\geq\; 2^{-2p_j(n)} \cdot 2^{2(p_j(n)-1)-1} \\
&= 2^{-3} \;=\; \frac{1}{8}.
\end{aligned}
$$

Next we shall show that $\sigma_2(h_1) \leq \frac{1}{16}$. Let $F_{x,h_1} = \mathrm{ran}(f_n^{(j)} \circ h_1)$. The cardinality of F_{x,h_1} is at most 2^l.

$$
\begin{aligned}
\sigma_2(h_1) &= \mathbf{Pr}_{h_2}[\exists x_1, x_2 \in F_{x,h_1}[x_1 \neq x_2 \wedge h_2(x_1) = h_2(x_2) = z] \mid h_2 \in H_{n,l+2}] \\
&\leq \sum_{x_1,x_2 \in F_{x,h_1} \wedge x_1 \neq x_2} \mathbf{Pr}_{h_2}[h_2(x_1) = h_2(x_2) = z \mid h_2 \in H_{n,l+2}] \\
&\leq \|F_{x,h_1}\|^2 \cdot 2^{-2(l+2)} \;\leq\; 2^{2l} \cdot 2^{-2(l+2)} \\
&= 2^{-4} \;=\; \frac{1}{16}.
\end{aligned}
$$

To complete the proof of the claim, we combine the above results.

$$
\sigma \geq \sigma_1 - \max_{h_1 \in H_{l,p_j(n)}} \sigma_2(h_1) \geq \frac{1}{8} - \frac{1}{16} \geq \frac{1}{16}.
$$

This completes the proof. ∎

To use Lemma 5.5.3, it suffices to show that, for each string w, $\hat{\eta}(\{(y,s) \mid y \in D \wedge w \in Q(N,B,y,s)\}) \leq \hat{\nu}_{\mathrm{stand}}(w)$ for some η which avrp-dominates μ. Let $d(n) = 24cn(n+1)q(n)^2$. Then, $2^{2\mathrm{llog}(n)+2\mathrm{ilog}(n)+2\mathrm{llog}(w)+5} \leq d(n)$. For this semi-distribution η, let us define $\hat{\eta}(x,s) = \frac{1}{d(|x|)\cdot 2p_{j_x}(|x|)} \cdot 2^{-|s|} \cdot \hat{\mu}(x)$ for all nonempty strings x. Notice that $\mathbf{Pr}_s[\langle e, s_n, s_i^n, s_k^m, h_1h_2z\rangle \in Q(N,y,s)] = 2^{-|h_1|-|h_2|}$. Let $w = \langle e, s_n, s_i^n, s_k^m, h_1h_2z\rangle$, and $m = p_{j_x}(n)$. Then, $2^{-p_j(n)+k} = 2^{-|z|+2}$.

$$
\begin{aligned}
&\hat{\eta}(\{(y,s) \mid y \in D \wedge \langle e, s_n, s_i^n, s_k^m, h_1h_2z\rangle \in Q(N,B,y,s)\}) \\
&\quad = \hat{\eta}(\langle x, s_j^{q(n)} s_k^{p_j(n)}\rangle, h_1h_2) \\
&\quad = \frac{1}{d(|x|)\cdot p_j(n)^2} \cdot \hat{\mu}(x) \cdot 2^{-|h_1|-|h_2|} \\
&\quad \leq \frac{1}{d(|x|)} \cdot 2^{-2\mathrm{ilog}(p_j(n))} \cdot 2^{-p_j(n)+k} \cdot 2^{-|h_1|-|h_2|} \\
&\quad = 2^{2\mathrm{llog}(n)+2\mathrm{ilog}(n)+2\mathrm{llog}(w)+5} \cdot \cdot 2^{-2\mathrm{ilog}(m)} \cdot 2^{-|z|+2} \cdot 2^{-|h_1h_2|} \\
&\quad = 2^{-4-2\mathrm{llog}(n)-2\mathrm{ilog}(n)-2\mathrm{ilog}(m)-|h_1h_2z|} \cdot 2^{-2\mathrm{llog}(w)-1} \\
&\quad \leq 2^{-|\langle e,s_n,s_i^n,s_k^m,h_1h_2z\rangle|-2\mathrm{llog}(w)-1} \;=\; 2^{-|w|-2\mathrm{llog}(w)-1} \\
&\quad = \hat{\nu}_{\mathrm{stand}}(w),
\end{aligned}
$$

since $|\langle e, s_n, s_i^n, s_k^m, h_1 h_2 z\rangle| \leq 2(2 + \mathrm{llog}(n) + \mathrm{ilog}(n) + \mathrm{ilog}(m)) + |h_1| + |h_2| + |z| + 1$. Notice that

$$\mathrm{llog}(w) \leq \log(|\langle e, s_n, s_i^n, s_k^m, h_1 h_2 z\rangle| + 1) \leq c \cdot (\log(p_j(n)) + \log n + 1) \leq c'(j + \log n + 1) \leq c'' q(n).$$

Finally we apply Lemma 5.5.3, and thus there exists a randomized Turing machine M_1 such that $\lambda x s.M_1(\langle x, h(x)\rangle; s)$ is polynomial on μ-average with the condition that, for each $s' \in \Gamma_{M_0}(x)$,

$$\mathbf{Pr}_s[M_1(\langle x, M_0(x; s')\rangle; s) = A(x)] \geq 1/16.$$

By the remark following Proposition 3.5.33, we immediately obtain that $(A, \mu) \in \mathrm{Aver}(\mathbf{BPP}, *)$. The proof is completed. □

The class $\mathbf{BPP}_{\mathbf{P}\text{-comp}}$ is closely related to $\mathbf{P}$-samplable distributions. In the following lemma, we see that an assumption like $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$ causes the $\mathbf{P}$-samplable distributions relative to $\mathbf{NP}$ oracles to be average $\mathbf{P}$-samplable.

Proposition 7.6.4 [96]

1. $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$ *implies* $\mathbf{P}_{\mathrm{tt}}^{\mathbf{NP}}\text{-samp} \subseteq^{\mathrm{p}} \mathbf{avP}\text{-samp}$.
2. $\mathbf{\Delta}_2^{\mathrm{p}} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$ *implies* $\mathbf{P}^{\mathbf{NP}}\text{-samp} \subseteq^{\mathrm{p}} \mathbf{avP}\text{-samp}$.

Proof. We shall show only claim (2). Assume that $\mathbf{\Delta}_2^{\mathrm{p}} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$. Take an arbitrary $\mathbf{P}^{\mathbf{NP}}$-samplable distribution μ. By its definition, there exists a sampling algorithm M, a polynomial q, and a set $A \in \mathbf{NP}$ such that, for any string x and any number $i \in \mathbb{N}$,

$$|\hat{\mu}(x) - \mathbf{Pr}_M[M^A(0^i) = x \text{ in time } q(|x|, i)]| \leq 2^{-i}.$$

We can assume without loss of generality that if M with oracle A on input 0^i halts and outputs x, then its running time does not exceed $q(|x|, i)$.

Let us define the set B as follows.

$$B = \{\langle 0^i, 1^n, s, 1^j, d\rangle \mid d \in \{0,1\} \wedge s \in \Sigma^{\leq q(n,i)} \wedge |M^A(0^i; s)| = n \\ \wedge \mathrm{Time}_M^A(0^i; s) \leq q(n, i) \wedge \text{ the } j\text{-th bit of } M^A(0^i; s) \text{ is } d\ \}.$$

Let $p(n) = 2n + 4$ for all $n \in \mathbb{N}$. Since $B \in \mathbf{\Delta}_2^{\mathrm{p}}$, our assumption ensures the existence of a probabilistic Turing machine N which accepts B in polynomial time on ν_{stand}-average with error probability $2^{-p(n)-n-i-4}$, where input is of the form $\langle 0^i, 1^n, s, 1^j, d\rangle$. Consider the following sampling machine M'.

begin sampling algorithm for M'
  input 0^i
  generate a natural number n_0 randomly
  generate a string s of length $\leq q(n_0, i + p(n))$
  for $n = n_0$ to ∞

generate s' at random so that $|ss'| \leq q(n, i+p(n))$
set $s := ss'$
for $j = 1$ to n **do**
if $N(\langle 0^{i+p(n)}, 1^n, s, 1^j, 0\rangle) = N(\langle 0^{i+p(n)}, 1^n, s, 1^j, 1\rangle)$ **then** go to $(*)$
if $N(\langle 0^{i+p(n)}, 1^n, s, 1^j, 0\rangle) < N(\langle 0^{i+p(n)}, 1^n, s, 1^j, 1\rangle)$
then set $d_j := 1$ **else** set $d_j := 0$
end for
output $d_1 d_2 \cdots d_n$ and **halt**
$(*)$ **end for**
end.

In the following analysis, we fix i and x. Let $|x| = n$. Set

$$S_{x,i} = \{s \in \Gamma_{M^A}(0^{i+p(|x|)}) \mid |s| \leq q(|x|, i+p(|x|)) \wedge M^A(0^{i+p(n)}; s) = x \text{ in time } q(|x|, i+p(|x|))\}.$$

Write σ_x^i to mean $\mathbf{Pr}_s[M^A(0^{i+p(n)}) = x \text{ in time } q(|x|, i+p(|x|)) \mid s \in \Omega_{M^A}(0^{i+p(n)})]$. By the definition of $S_{x,i}$, we have $\sigma_x^i = \sum_{s \in S_{x,i}} 2^{-|s|}$.

Note that the success probability that each iteration of the second **for**-loop is at least:

$$\begin{aligned}
&\mathbf{Pr}_N[N(\langle 0^{i+p(n)}, 1^n, s, 1^j, \overline{x_j}\rangle) < \langle 0^{i+p(n)}, 1^n, s, 1^j, x_j\rangle)] \\
&\quad = \mathbf{Pr}_N[N(\langle 0^{i+p(n)}, 1^n, s, 1^j, \overline{x_j}\rangle) = 0 \wedge N(\langle 0^{i+p(n)}, 1^n, s, 1^j, x_j\rangle) = 1] \\
&\quad \geq (1 - 2^{-p(n)-n-i-4})^2 \\
&\quad \geq 1 - 2^{-p(n)-n-i-1}.
\end{aligned}$$

Now let us denote by ρ_x^i the probability that the machine M' on input $0^{i+p(n)}$ with random input $s \in S_{x,i}$ outputs x. The function ρ_x^i clearly does not exceed σ_x^i. The lower bound of ρ_x^i is calculated as follows:

$$\begin{aligned}
\rho_x^i &= \sum_{s \in S_{x,i}} 2^{-|s|} \cdot \prod_{j=1}^{n} \mathbf{Pr}_N[N(\langle 0^{i+p(n)}, 1^n, s, 1^j, \overline{x_j}\rangle) < N(\langle 0^{i+p(n)}, 1^n, s, 1^j, x_j\rangle)] \\
&\geq \sum_{s \in S_{x,i}} 2^{-|s|} \cdot \prod_{j=1}^{n} (1 - 2^{-p(n)-n-i-1}) \quad = \quad \sigma_x^i \cdot (1 - 2^{-p(n)-n-i-1})^n \\
&\geq (1 - 2^{-p(n)-i}) \cdot \sigma_x^i.
\end{aligned}$$

The last inequality follows from Lemma A.6.

We then have

$$\left(1 - \frac{1}{2^{i+p(n)}}\right) \cdot \sigma_x^i \leq \rho_x^i \leq \sigma_x^i.$$

Therefore, we obtain $|\rho_x^i - \rho_x^j| \leq 2^{-p(n)+1}(2^{-i} + 2^{-j})$, a result analogous to Theorem 4.4.13.

Let $\overline{\rho}_x^i$ be the probability that the machine M' on input 0^i outputs x. We then have:

$$\frac{1 - 2^{-i-p(n)}}{8(n+1)^2} \cdot \sigma_x^i \leq 2^{-2\mathrm{llog}(n)-1} \cdot \rho_x^i \leq \overline{\rho}_x^i \leq \sum_{k=0}^{n} 2^{-2\mathrm{llog}(k)-1} \cdot \rho_x^i < 2 \cdot \sigma_x^i.$$

As in Theorem 4.4.13, we can show that $|\overline{\rho}_x^i - \overline{\rho}_x^j| \leq 2^{-i} + 2^{-j}$ for all $i, j > 0$.

It is not difficult to show that N' runs in polynomial time on μ-average with respect to the size of its output. Therefore, μ belongs to **avP**-samp. □

Corollary 7.6.5 [96]

1. $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$ *implies* $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}_{tt}^{NP}\text{-samp}}$.
2. $\mathbf{\Delta}_2^p \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$ *implies* $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}^{NP}\text{-samp}}$.

Proof. Assume that $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{P}\text{-comp}}$. Notice that $\mathbf{P}_{tt}^{\mathbf{NP}}$-samp $\subseteq^p$ **avP**-samp implies $\mathbf{BPP}_{\mathbf{avP}\text{-samp}} \subseteq \mathbf{BPP}_{\mathbf{P}_{tt}^{NP}\text{-samp}}$. By Proposition 7.6.4, it follows that $\mathbf{NP} \subseteq \mathbf{BPP}_{\mathbf{avP}\text{-samp}} \subseteq \mathbf{BPP}_{\mathbf{P}_{tt}^{NP}\text{-samp}}$.

A similar argument leads to (2). □

7.7 Random Oracle Separations

We return to Levin's original question of whether $\mathbf{NP} \subseteq \mathbf{P}_{\mathbf{P}\text{-comp}}$. Since this question is difficult to answer, we turn our interest to its relativization.

Bennett and Gill [8] first used a notion of "random oracles" and made several important contributions to computational complexity theory including the result that, relative to a random oracle, three classes **P**, **NP**, and co-**NP** are different. Intuitively, if we choose an oracle set A at random, $\mathbf{P}^A$ is different from $\mathbf{NP}^A$ with probability 1. In other words, "most" oracles can separate **P** from **NP**. In the same paper, Bennett and Gill proposed a "random oracle hypothesis" that states: if a property $\mathcal{P}$ holds relative to a random oracle, then $\mathcal{P}$ also holds in the non-relativized world. (However, this hypothesis is now known to be false [57].)

To approach Levin's original question, we consider randomly relativized world. We remark that there is no known inclusion relationship between the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ and other worst-case complexity classes, such as **NP** and **PSPACE**. This section will discuss, relative to a random oracle, several negative results about these relationships.

We identify a set A with a binary real number $0.r$, where $r = \chi_A(0)\chi_A(1)\chi_A(00)\chi_A(01)\cdots$.

Definition 7.7.1 (Random Oracles) Let $\mathcal{Q}^X$ be a property relativized to oracle X. We say that, *(with probability 1) $\mathcal{Q}^X$ holds relative to a random oracle X* if the Lebesgue measure of the set $\{X \mid \mathcal{Q}^X \text{holds}\}$ is 1, denoted by $\mathbf{m}(\{X \mid \mathcal{Q}^X \text{ holds }\}) = 1$.

In our setting, the Lebesgue measure behaves like a probability measure if the property $\mathcal{Q}^X$ is finitely evaluated. Here is an example. Let

$$\eta_A(x) = A(x10)A(x100)A(x1000)\cdots A(x10^{|x|}).$$

Then, clearly $|\eta_A(x)| = |x|$ for all x. Consider the event $\{A \mid \eta_A(x) = 0^{|x|}\}$ for each fixed string x. The measure of this event, $\mathbf{m}(\{A \mid \eta_A(x) = 0^{|x|}\})$, is $2^{-|x|}$. Hence, for each $n \in \mathbb{N}$, $\mathbf{m}(\{A \mid \forall y(\eta_A(y) \neq 0^n)\}) = (1-2^{-n})^{2^n}$, which approaches $1/e$ as n goes to ∞.

We shall show that, with probability 1, $\mathbf{NP}_{\mathbf{P}\text{-comp}}$ is different from $\mathbf{P}_{\mathbf{P}\text{-comp}}$ relative to a random oracle. In this section, we choose the following relativization of the classes $\mathbf{P}_{\mathcal{F}}$ and $\mathbf{NP}_{\mathcal{F}}$.

Definition 7.7.2 (Relativization) Let X be a set of strings, and let $\mathcal{F}^X$ be a set of distributions relative to X.

1. Let $\mathbf{P}^X_{\mathcal{F}^X}$ be the collection of all sets A such that, for any distribution μ in $\mathcal{F}^X$, (A,μ) belongs to $\mathrm{Aver}(\mathbf{P},\mathcal{F})^{(X,\nu)}$ for some distribution ν.

2. Let $\mathbf{NP}^X_{\mathcal{F}^X}$ be the collection of all sets A such that, for any distribution μ in $\mathcal{F}^X$, (A,μ) belongs to $\mathrm{Aver}(\mathbf{NP},\mathcal{F})^{(X,\nu)}$ for some distribution ν.

Proposition 7.7.3 **[8]** *Relative to a random oracle X,* $\mathrm{TALLY} \cap \mathbf{NP}^X \not\subseteq \mathbf{P}^X$.

Proof. We can even show the slightly stronger statement that $\mathrm{TALLY} \cap \mathbf{NP}^X \not\subseteq \text{co-}\mathbf{NP}^X$ relative to a random oracle X. This clearly yields the desired consequence. In what follows, we shall prove this stronger statement.

We first introduce a test language which lies in $\mathrm{TALLY} \cap \mathbf{NP}^X$ for any oracle set X. For this purpose, let us define $RANGE^A = \{0^n \mid \exists y[\eta_A(y) = 0^n]\}$. Clearly $RANGE^A$ is in TALLY and also in $\mathbf{NP}^A$ for any oracle A. Let M be an arbitrary polynomial-time oracle Turing machine. Let $C^0_n = \{A \mid \forall y[\eta_A(y) \neq 0^n]\}$ and let $C^1_n = \{A \mid \eta_A(0^n) \neq 0^n \wedge \exists! y(\eta_A(y) = 0^n)\}$. As seen above, $\mathbf{m}(C^0_n) = (1-2^{-n})^{2^n}$ for any natural number n, and consequently $\lim_{n\to\infty} \mathbf{m}(C^0_n) = 1/e$, where e is the base of the natural logarithms.

Let $\mathcal{Y}$ be the probability space of $\Sigma^n - \{0^n\}$. We introduce a transformation from $C^0_n \times \mathcal{Y}$ to $C^1_n \times \mathcal{Y}$ as $f(A,y) = (A_y, \eta_A(y))$, where $A_y = A - \{y10^m \mid 1 \leq m \leq n\}$. We now claim that f is surjective. For any $(B,u) \in C^1_n \times \mathcal{Y}$, there exists a unique y such that $\eta_B(y) = 0^n$. Let $A = B \cup \{y10^m \mid 1 \leq m \leq n\}$. Then, $\eta_A(z) \neq 0^n$ for all z because $\eta_A(y) \neq 0^n$.

Moreover, for any event $E \subseteq C^1_n$, we have

$$\mathbf{m}(\{A \mid A \in E\}) = \sum_{y \in \mathcal{Y}} \frac{1}{2^n - 1} \cdot \mathbf{m}(\{A \mid A \in C^0_n \wedge A_y \in E\}).$$

Hence, in particular, it holds that

$$\mathbf{m}(C^1_n) = \sum_{y \in \mathcal{Y}} \frac{1}{2^n - 1} \cdot (C^0_n) = \mathbf{m}(C^0_n)$$

because the term $\mathbf{m}(C^0_n)$ does not depend on the choice of y.

Let us define two conditional probabilities. Let

$$\alpha^0_n = \frac{\mathbf{m}(\{A \mid 0^n \in L(M,A) \wedge A \in C^0_n\})}{\mathbf{m}(C^0_n)}; \text{ and}$$

$$\alpha_n^1 = \frac{\mathbf{m}(\{A \mid 0^n \in L(M,A) \wedge A \in C_n^1\})}{\mathbf{m}(C_n^1)}.$$

Now we shall claim that $\alpha_n^1 \geq \frac{99}{100} \cdot \alpha_n^0$ for almost all n.

Claim 19 *$\alpha_n^1 \geq \frac{99}{100} \cdot \alpha_n^0$ holds for almost all n.*

Proof of Claim. Let n be large enough that $\frac{p(n)}{2^{p(n)}} \leq \frac{1}{100}$. For any pair $(A,z) \in C_n^0 \times \mathcal{Y}$, with conditional probability α_0, M^A accepts 0^n. Let us fix one of its accepting paths and denote it by p. Let us consider the oracle set A_z. On the path p, the probability that M does not make all queries of the form $z10^k$, $1 \leq k \leq n$, is at least $1 - \frac{p(n)}{2^n} \geq \frac{99}{100}$. With this probability, M^{A_z} accepts 0^n. Thus, α_n^1 can be estimated as follows:

$$\begin{aligned}
\alpha_n^1 &= \frac{\mathbf{m}(\{A \mid 0^n \in L(M,A) \wedge A \in C_n^1\})}{\mathbf{m}(C_n^1)} \\
&= \sum_{y \in \mathcal{Y}} \frac{1}{2^n - 1} \cdot \frac{\mathbf{m}(\{A \mid 0^n \in L(M,A) \wedge A \in C_n^0 \wedge 0^n \in L(M,A_y)\})}{\mathbf{m}(C_n^1)} \\
&\geq \frac{99}{100} \cdot \frac{\mathbf{m}(\{A \mid 0^n \in L(M,A) \wedge A \in C_n^0\})}{\mathbf{m}(C_n^0)} \\
&= \frac{99}{100} \cdot \alpha_n^0.
\end{aligned}$$

■

Finally we must calculate the overall error probability $\epsilon = \mathbf{m}(\{A \mid M^A(0^n) \neq \overline{RANGE^A}(0^n)\})$. Choose an integer n large enough that we can guarantee that $\mathbf{m}(C_n^0) \geq \frac{36}{100}$ and $\alpha_n^1 \geq \frac{99}{100}\alpha_n^0$. For this n,

$$\begin{aligned}
\epsilon &\geq (1 - \alpha_n^0) \cdot \mathbf{m}(C_n^0) + \alpha_n^1 \cdot \mathbf{m}(C_n^1) \\
&\geq (1 - \alpha_n^0) \cdot \mathbf{m}(C_n^0) + \frac{99}{100}\alpha_n^0 \cdot \mathbf{m}(C_n^0) \\
&= \left(1 - \alpha_n^0 + \frac{99}{100} \cdot \alpha_n^0\right) \cdot \mathbf{m}(C_n^0) \\
&\geq \left(1 - \frac{1}{100}\alpha_n^0\right) \cdot \frac{36}{100} \quad \geq \quad \left(1 - \frac{1}{100}\right) \cdot \frac{36}{100} \\
&> \frac{1}{3}.
\end{aligned}$$

Hence, the error probability ϵ is not 0. Thus, the event that M^A computes the complement of $RANGE^A$ has measure 0 by the $0-1$ Law. In other words, $RANGE^X \notin \text{co-}\mathbf{NP}^X$ relative to a random oracle X. □

Using the above proposition, we can show the following separation result.

Proposition 7.7.4 *Relative to a random oracle X, $\mathbf{P}^X_{\mathbf{P}^X\text{-comp}} \neq \mathbf{NP}^X_{\mathbf{P}^X\text{-comp}}$.*

Proof. For any oracle A, $\mathbf{P}^A_{\mathbf{P}^A\text{-comp}} = \mathbf{NP}^A_{\mathbf{P}^A\text{-comp}}$ clearly implies $\text{TALLY} \cap \mathbf{NP}^A \subseteq \mathbf{P}^A$. By Proposition 7.7.3, relative to a random oracle X, $\text{TALLY} \cap \mathbf{NP}^X \not\subseteq \mathbf{P}^X$ with probability 1. Hence, we obtain the desired result. □

Proposition 7.7.5 **[8]** *Relative to a random oracle X, there exists a $\mathbf{P}^X$-bi-immune set in $\mathbf{NP}^X$.*

Proof. For any oracle A, we define $RANGE_3^A$ as the set $\{x \mid \exists y[\eta_A(y) = xxx]\}$. Clearly $RANGE_3^A$ is in $\mathbf{NP}^A$. In the following, we want to prove that $RANGE_3^X$ is $\mathbf{P}^X$-bi-immune relative to a random oracle X.

First we claim that $RANGE_3^X$ is $\mathbf{P}^X$-immune relative to a random oracle X. Let us fix a polynomial-time deterministic oracle Turing machine M and let p be an increasing polynomial which bounds the running time of M. If $L(M, X)$ is finite relative to a random oracle X, then the claim is trivially true. Now we assume otherwise.

We say that y is *examined* by M on input w if M^A on input w queries a string of the form $y10^m$, $1 \leq m \leq |y|$. Let

$$EXAM^A(w) = \{y \mid M^A \text{ on } w \text{ examines } y, \text{ but there is no } v < w \text{ such that } M^A \text{ on } v \text{ examines } y\ \}.$$

Using the set $EXAM^A(w)$, we define $EVID^A$ as follows:

$$EVID^A = \bigcup_{w \in \Sigma^*} \{y \mid y \in EXAM^A(w) \wedge \exists x \geq w[\eta_A(y) = xxx]\}.$$

Now we shall show that, relative to a random oracle X, $EVID^X$ is finite.

Claim 20 *$EVID^X$ is finite relative to a random oracle X.*

Proof of Claim. Note that, by the polynomial bound on M, $\|EXAM^A(w)\| \leq p(|w|)$ for any string w. Now fix any sufficiently large string w satisfying $\|EXAM^A(w)\| \leq 2^{|w|/2}$ for all oracles A. For each y,

$$\mathbf{m}(\{A \mid \exists x \geq w[\eta_A(y) = xxx]\}) \leq 2^{|w|} \cdot 2^{-3|w|} = 2^{-2|w|}.$$

For simplicity, we write $F^A(w)$ to mean that there exist two strings x and y such that $x \geq w$, y in $EXAM^A(w)$, and $\eta_A(y) = xxx$. The probability that $F^A(w)$ holds is at most

$$\mathbf{m}(\{A \mid F^A(w)\}) \leq 2^{|w|/2} \cdot 2^{-2|w|} = 2^{-3|w|/2}.$$

Hence, $\sum_w \mathbf{m}(\{A \mid F^A(w) \text{ holds }\}) \leq \sum_w 2^{-3|w|/2} < \infty$. Thus, by Borel-Cantelli's Lemma (Lemma A.11), $\mathbf{m}(\{A \mid \overset{\infty}{\exists}\, w F^A(w)\}) = 0$. This implies that $\mathbf{m}(\{A \mid EVID^A \text{ is infinite }\}) = 0$. ■

Let us define $D = \{A \mid L(M, A) \text{ is an infinite subset of } RANG_3^A \text{ and } EVID^A \text{ is finite }\}$. To lead to a contradiction, it suffices to show that $\mathbf{m}(D) = 0$. To do so, for each set A, we shall define the series of strings $\{x_i^A\}_{i \in \mathbb{N}}$ as follows: initially, set $x_0^A = \lambda$, and for each $k \geq 1$, set

$$x_k^A = \begin{cases} \min\{x \in L(M, A) \mid x > x_{k-1}^A \wedge \forall y \in EVID^A[\eta_A(y) \neq xxx]\}, & \text{if one exists,} \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

Note that x_k^A may not always be defined. Using this series, we define

$$D_k = \{A \mid x_k^A \text{ exists and } \forall i < k[x_i^A \in RANG_3^A]\}$$

for $k \geq 1$. It is not difficult to see that $D_1 \supseteq D_2 \supseteq \cdots \supseteq D$.

We next claim that, for almost all k, the ratio $\mathbf{m}(D_{k+1})/\mathbf{m}(D_k)$ does not exceed some constant c less than 1.

Claim 21 *There exists a constant c in the interval $[0,1)$ such that $\mathbf{m}(D_{k+1}) \leq c \cdot \mathbf{m}(D_k)$ for almost all $k \in \mathbb{N}$.*

Proof of Claim. Take any sufficiently large integer k_0 such that $p(k) \cdot 2^{k+1} \leq 2^{2k}$ holds for all integers $k \geq k_0$. Fix any such integer k.

Since $\| \bigcup_{w \leq x} EXAM^A(w) \| \leq p(|x|) \cdot 2^{|x|+1} \leq 2^{2|x|}$ for each x with $|x| \leq k_0$, it holds that

$$\forall y \in \bigcup_{w \leq x} EXAM^A(w)[\eta_A(y) \neq xxx] \wedge \exists y \notin \bigcup_{w \leq x} EXAM^A(w)[\eta_A(y) = xxx].$$

Since $D_{k+1} = D_{k+1} \cap D_k$, the ratio $\mathbf{m}(D_{k+1})/\mathbf{m}(D_k)$ is equal to the conditional probability $\mathbf{m}(D_{k+1} \mid D_k)$. This is bounded by the probability that there exists a string x accepted by M with oracle A such that $\eta_A(y) \neq xxx$ for all y in $EXAM^A(w)$ for some $w \leq x$, but $\eta_A(y') = xxx$ for some y' which is not in $\bigcup_{w \leq x} EXAM^A(w)$, provided that x_k^A exists. Hence,

$$\mathbf{m}(D_{k+1} \mid D_k) \leq \mathbf{m}(\{A \mid \exists x > x_k^A \exists y[\eta_A(y) = xxx]\}).$$

The last expression tends to $1 - 1/e$ as k approaches ∞. ■

Therefore, for almost all k, $\mathbf{m}(D_{k+1}) \leq c \cdot \mathbf{m}(D_k)$, which implies $\mathbf{m}(D) = \lim_{k \to \infty} c^k$. This yields the conclusion $\mathbf{m}(D) = 0$.

The proof that $\overline{RANGE_3^X}$ is $\mathbf{P}^X$-immune relative to a random oracle proceeds similarly. □

The next theorem follows from the previous proposition.

Theorem 7.7.6 *Relative to a random oracle X, $\mathbf{NP}^X \not\subseteq \mathbf{P}^X_{\mathbf{P}^X\text{-comp}}$.*

Proof. By Proposition 7.7.5, there exists a $\mathbf{P}^X$-bi-immune set in $\mathbf{NP}^X$ for a random oracle X. Since Proposition 7.4.5 is relativizable, $\mathbf{P}^A_{\mathbf{P}^A\text{-comp}}$ has no $\mathbf{P}^A$-bi-immune sets for any oracle A. Therefore, $\mathbf{NP}^X$ cannot be included in $\mathbf{P}^X_{\mathbf{P}^X\text{-comp}}$ relative to a random oracle X. □

Theorem 7.7.7 *Relative to a random oracle X, $\mathbf{P}^X_{\mathbf{P}^X\text{-comp}} \not\subseteq \mathbf{PSPACE}^X$.*

Proof. The proof presented here is a modification of the proof of the corollary of Theorem 2 in [8].

In the proof, we identify strings with natural numbers; that is, if x is s_n, then x represents the natural number n. By this identification, it follows that $2^{|x|-1} + 3 \leq x \leq 2^{|x|} + 2$ for all $x \geq 4$. Let $\eta_A = A(x10)A(x10^2) \cdots A(x10^{|x|})$ and $QUERY^A = \{x \mid x \text{ is a candidate}, \eta_A(0^x) \in A\}$. It follows that $QUERY^A \in \mathbf{P}^A_{\mathbf{P}^A\text{-comp}}$ for all oracles A.

Take any polynomial-space deterministic Turing machine M. For each x, let $C_x = \{A \mid \eta_A(0^x) \notin Q(M, A, x)\}$ and $\overline{C}_x = \{A \mid A \notin C_x\}$. We shall show that $\mathbf{m}(C_x)$ is not small.

Claim 22 $\mathbf{m}(C_x) \geq 1/3$ *for almost all strings* x.

Proof of Claim. Notice that the tape space used by the machine M on input x is bounded above by a polynomial in $|x|$. Let p be such a polynomial. By the space bound, the number of all possible instantaneous descriptions of M on input x is at most $2^{p(|x|)}$, and this number is irrelevant to the choice of oracles. This number is also an upper bound of the number of queries made by M. On the other hand, to determine $\eta_A(0^x)$, we need to check at most $2^{|x|-1}$ bits since

$$|\eta_A(0^x)| = x \geq 2^{|x|-1} + 3 \geq 2^{|x|-1}.$$

The measure of $\overline{C}_x$ is at most the ratio of the number of query strings to the number of bits to determine $\eta_A(0^x)$, and hence

$$\mathbf{m}(\overline{C}_x) \leq \frac{\| \bigcup_A Q(M, A, x)\|}{2^{\min_A |\eta_A(0^x)|}} \leq \frac{2^{p(|x|)}}{2^{2^{|x|-1}}}.$$

Obviously, $\mathbf{m}(\overline{C}_x)$ approaches 0 when x grows, and therefore $\lim_{x\to\infty} \mathbf{m}(C_x) = 1$. ■

Next let us consider the error probability $\epsilon = \mathbf{m}(\{A \mid M^A(x) \neq QUERY(x)\})$. We must show that $\epsilon \geq \frac{1}{3}$. Note that

$$\epsilon \geq \mathbf{m}(\{A \in C_x \mid x \in L(M, A) \wedge \eta_A(0^x) \notin A\}) + \mathbf{m}(\{A \in C_x \mid x \notin L(M, A) \wedge \eta_A(0^x) \in A\}).$$

To obtain the desired lower bound of ϵ, we define two sets:

$$\begin{aligned} C_x^0 &= \{A \in C_x \mid x \in L(M, A) \wedge \eta_A(0^x) \in A\}, and \\ C_x^1 &= \{A \in C_x \mid x \in L(M, A) \wedge \eta_A(0^x) \notin A\}. \end{aligned}$$

Next we shall show that $\mathbf{m}(C_x^0) = \mathbf{m}(C_x^1)$. To show this, we define the transformation f_x as

$$f_x(A) = \begin{cases} A - \{\eta_A(0^x)\} & \text{if } \eta_A(0^x) \in A, \\ A \cup \{\eta_A(0^x)\} & \text{otherwise.} \end{cases}$$

It is easy to see that f_x is bijective on C_x. Hence, from the fact that $\mathbf{m}(\{A \in C_x \mid x \in L(M, A)\}) = \mathbf{m}(C_x^0 \cup C_x^1)$, it follows that $\mathbf{m}(C_x^0) = \frac{1}{2} \cdot \mathbf{m}(\{A \in C_x \mid x \in L(M, A)\})$.

A similar argument shows that $\mathbf{m}(\{A \in C_x \mid x \notin L(M, A) \wedge \eta_A(0^x) \in A\}) = \frac{1}{2} \cdot \mathbf{m}(\{A \in C_x \mid x \notin L(M, A)\})$. By combining the two equations, we obtain

$$\begin{aligned} \epsilon &\geq \frac{1}{2} \cdot \mathbf{m}(\{A \in C_x \mid x \in L(M, A)\}) + \frac{1}{2} \cdot \mathbf{m}(\{A \in C_x \mid x \notin L(M, A)\}) \\ &= \frac{1}{2} \cdot \mathbf{m}(C_x). \end{aligned}$$

The last term exceeds $\frac{2}{3}$ for any large string x. Therefore, $\epsilon \geq \frac{1}{3}$. □

Chapter 8

Conclusion

In the early days of average-case analysis, researchers had great hopes of solving all **NP**-complete problems in average polynomial time. The satisfiability problem, for example, can be solved in average polynomial time with respect to some natural input distribution, and so have the Hamiltonian circuit problem with some input distribution. As research has progressed, however, the realm of average-case analysis has encountered the same difficulty as its worst-case counterpart.

L. Levin showed that there are problems intractable even in the average-case setting. Since Levin presented his approach to average-case complexity theory, much research has been devoted to grasping what the intractability of problems is. These average-case analyses have attracted significant attention from researchers in the other fields, such as cryptography and statistical physics. Researchers have continued to seek for another complete problem for Dist(**NP**, **P**-comp). At the same time, there have been a number of different approaches developed to gain a better understanding of average-case *intractable* problems. One of them is to study the average-case complexity of distributional search problems. For example, **NP** search problems have recently been shown not to be harder than **NP**-problems in the average-case setting.

This thesis tried to contribute to the development of a general and consistent theory of average-case complexity. Personally, I have been inspired by Levin's early question of whether all **NP**-complete problems are solvable in polynomial time on the average with respect to naturally selected distributions. This question is deeply related to the **E** =?**NE** question as well as the **P** =?**NP** question in worst-case complexity theory. We thus see a tie between average-case complexity theory and worst-case complexity theory.

Reducibilities have played a central role in our study of the structural properties of those classes. One of the significant features of this thesis is the introduction of two average-case versions of the polynomial-time hierarchy, one by average-case Turing reducibilities, and the other by a model of alternating Turing machines. These two hierarchies preserve numerous properties of the worst-case hierarchy, but they are intrinsically different in character because of their sensitivity to the choice of input distributions. A structural study of average classes has commenced in recent years, whereas there have already been a number of studies on worst-case complexity classes.

Another important feature of this thesis is to introduce the notion of quintessential complexity classes.

This notion actually enables us to bridge the gap between average-case complexity theory and its worst-case counterpart. In particular, the class $\mathbf{P}_{\mathbf{P}\text{-comp}}$ has been thoroughly studied in this thesis. We have seen a variety of possibilities in this direction that contribute to an understanding of average-case complexity.

We regret that there remain many intriguing topics in average-case complexity theory, and we have left numerous questions unsolved in this thesis. One important direction is the study of distributional search problems. There are a number of studies in this area (see e.g., [9, 12, 113, 108]).

Average-case complexity theory is a fruitful field to cultivate. However, it is not always easy to seek the right definition of average-case counterparts of well-known concepts in worst-case complexity theory. For example, we have already seen several possible ways of defining the class $\mathrm{Aver}(\mathbf{NP}, \mathcal{F})$. Consider the following example. Suppose that we wish to ask ourselves: "Is it possible to define in a *natural way* an average-case version of $\#\mathbf{P}$?" Recall that the $\#\mathbf{P}$-functions were originally introduced by the model of polynomial-time counting machines (i.e., nondeterministic Turing machines which sum the number of accepting configurations). Our definition of nondeterministic Turing machines does not seem to provide any reasonable model for counting machines because not all accepting computation paths are considered in measuring the running time of the machines. One possible way to define such a class, say $\mathrm{Aver}(\#\mathbf{P}, \mathcal{F})$, is given as follows:

Definition 8.0.8 (Average #P Functions) A function f on Σ^* is called $\#\mathbf{P}$-*computable on* μ-*average* if there exists a randomized Turing machine M such that (i) M is polynomial-time bounded on μ-average, and (ii) $f(x)$ is the number of accepting computation paths of M on input x. The class $\mathrm{Aver}(\#\mathbf{P}, \mathcal{F})$ is defined to be the collection of all pairs (f, μ) such that f is a function which is $\#\mathbf{P}$-computable on μ-average.

R. Impagliazzo recently expressed the view in [43] that there are five possible worlds we might inhabit: *Algorithmica*, in which no intractable problems exist (i.e., $\mathbf{P} = \mathbf{NP}$ or $\mathbf{NP} \subseteq \mathbf{BPP}$ holds); *Heuristica*, in which there are intractable problems, but no problems are hard on the average (i.e., $\mathbf{P} \neq \mathbf{NP} \subseteq \mathbf{P}_{\mathbf{P}\text{-samp}}$); *Pessiland*, in which there are hard distributional problems, but no (strong) one-way functions exist; *Minicrypt*, in which one-way functions exist, but public key cryptography is impossible; and *Cryptomania*, in which public key cryptography is possible.

No matter which world we inhabit, researchers will continue to pursue attempts to understand natural phenomena.

Appendix A

Small Lemmas

This section provides several important lemmas used in this thesis.

The following inequality is known as Markov's inequality.

Lemma A.1 **(Markov's Inequality)** *Let μ be a distribution and let f be a function from Σ^* to $\mathbb{R}^+$. For every positive real number r, $\hat{\mu}(\{x \mid f(x) > r \cdot \mathrm{E}[f(X)]\}) < \frac{1}{r}$, where $\mathrm{E}[f(X)] = \sum_x f(x)\hat{\mu}(x)$.*

Proof. If $\mathrm{E}[f(X)] = 0$, then either $f(x) = 0$ or $\hat{\mu}(x) = 0$ for all $x \in \Sigma^*$. Thus, if $f(x) > 0$, then $\hat{\mu}(x)$ must be 0. This yields the consequence that $\hat{\mu}(\{x \mid f(x) > 0\}) = 0 < \frac{1}{r}$ for all $r > 0$.

Now we assume that $\mathrm{E}[f(X)] > 0$. Let $A = \{x \mid f(x) > r \cdot \mathrm{E}[f(X)]\}$. Since the case $A = \emptyset$ is trivial, we assume otherwise. Define $t_f(x) = 1$ if $f(x) > r \cdot \mathrm{E}[f(X)]$, and 0 otherwise. Since $t_f(x) < \frac{f(x)}{r \cdot \mathrm{E}[f(X)]}$ for all $x \in A$, we have

$$\hat{\mu}(A) = \mathrm{E}[t_f(X)] < \mathrm{E}\left[\frac{f(X)}{r \cdot \mathrm{E}[f(X)]}\right] = \frac{\mathrm{E}[f(X)]}{r \cdot \mathrm{E}[f(X)]} = \frac{1}{r}.$$

□

Lemma A.2 **(Jensen's Inequality)** *Let f be a strictly increasing concave function defined on an interval $(0, \infty)$ and set $f(\infty) = \lim_{x \to \infty} f(x)$. Then,*

$$E[f(X)] \leq f(E[X]),$$

where X is a random variable with values in $(0, \infty]$.

Proof. In the case where $E[X] = \infty$, we get

$$f(E[X]) = \lim_{x \to \infty} f(x) \geq E[f(X)].$$

On the contrary, suppose that $E[X] < \infty$. Since f is concave,

$$(*) \qquad \frac{f(v) - f(u)}{v - u} \geq \frac{f(w) - f(v)}{w - v}$$

for all positive real numbers u, v, and w satisfying $u < v < w$.

Let us fix v and consider the upper and lower limits:

$$\begin{aligned} g^*(v) &= \lim_{z\uparrow v} \frac{f(v)-f(z)}{v-z}; \quad \text{and} \\ g_*(v) &= \lim_{z\downarrow v} \frac{f(z)-f(v)}{z-v}. \end{aligned}$$

By the monotonicity of the function f, these limits $g^*(v)$ and $g_*(v)$ exist for any v in the interval $(0,\infty)$. The definition also implies $g^*(v) \geq g_*(v)$.

Claim 23 *For all $x, v \in (0,\infty)$ and every $c \in [g_*(v), g^*(v)]$, $f(x) \leq c(x-v) + f(v)$.*

Proof of Claim. If $x = v$, then the claim is trivial. Now suppose $x > v$. Then, by $(*)$, it follows that

$$\frac{f(x)-f(v)}{x-v} \leq \lim_{z\downarrow v} \frac{f(z)-f(v)}{z-v} = g_*(v) \leq c.$$

Hence, $f(x) - f(v) \leq c \cdot (x-v)$.

In the other case where $x < v$, we get

$$\frac{f(v)-f(x)}{v-x} \geq \lim_{z\uparrow v} \frac{f(v)-f(z)}{v-z} = g^*(v) \geq c,$$

and thus, we conclude that $f(x) - f(v) \leq c \cdot (x-v)$. ■

As a special case of the above claim, for a random variable X, $f(X) \leq c \cdot (X - E[X]) + f(E[X])$. By taking expectations,

$$E[f(X)] \leq c \cdot (E[X] - E[E[X]]) + E[f(E[X])] = f(E[X]).$$

□

Lemma A.3 *1.* $\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$.

2. $\sum_{n=1}^{\infty} \frac{1}{n^4} = \frac{\pi^4}{90}$.

3. $\sum_{n=1}^{\infty} \frac{1}{n^6} = \frac{\pi^6}{945}$.

Lemma A.4 *Let g be a function from $\Sigma^* \times \Sigma^*$ to $\mathbb{R}^+$. Let $x \in \Sigma^*$. The following two statements for g are equivalent.*

1. *There exists a function h such that $|h(x) - g(x,0^i)| \leq 2^{-i}$ for almost all x and $i \in \mathbb{N}$.*
2. *$|g(x,0^i) - g(x,0^j)| \leq 2^{-i} + 2^{-j}$ for almost all $i, j \in \mathbb{N}$.*

Proof. We first show that (1) implies (2). This is somewhat straightforward.

$$|g(x,0^i) - g(x,0^j)| = |g(x,0^i) - h(x) + h(x) - g(x,0^j)|$$

$$\begin{aligned} &\leq |g(x,0^i) - h(x)| + |h(x) - g(x,0^j)| \\ &\leq 2^{-i} + 2^{-j}. \end{aligned}$$

Conversely, we show that (2) implies (1). Assume that $|g(x,0^i) - g(x,0^j)| \leq 2^{-i} + 2^{-j}$ for almost all $i, j \in \mathbb{N}$. We first note that, for each x, there exists the limit $\lim_{j\to\infty} g(x,0^j)$ by a classical argument in analysis. We then set $h(x) = \lim_{i\to\infty} g(x,0^i)$.

Then,

$$|h(x) - g(x,0^i)| \leq |\lim_{k\to\infty} g(x,0^k) - g(x,0^i)| = \lim_{k\to\infty} |g(x,0^k) - g(x,0^i)| \leq \lim_{k\to\infty} (2^{-k} + 2^{-i}) = 2^{-i}.$$

□

Lemma A.5 *Assume that $\{a_i\}_{1\leq i\leq n}$ and $\{b_i\}_{1\leq i\leq n}$ are sequences of real numbers such that, for each i, $|a_i| \leq 1$ and $|b_i| \leq 1$. Then, $|\prod_{k=1}^n a_i - \prod_{k=1}^n b_i| \leq \sum_{k=1}^n |a_i - b_i|$.*

Proof. By induction on n, we shall show that

$$\left|\prod_{k+1}^{n} a_i - \prod_{k=1}^{n} b_i\right| \leq \sum_{k=1}^{n} \left[\left(\prod_{i=1}^{k-1} |a_i|\right) |a_k - b_k| \left(\prod_{j=k+1}^{n} |b_j|\right)\right],$$

where $\prod_{i=1}^{k-1} |a_i| = 1$ if $k = 1$, and $\prod_{j=k+1}^{n} |b_j| = 1$ if $k = n$. The lemma immediately follows from this inequality.

The base case $n = 1$ is trivial. Assume that $n \geq 1$. Since $|aA - bB| \leq |a| \cdot |A - B| + |a - b| \cdot |B|$ holds in general, we have

$$\begin{aligned} \left|\prod_{k=1}^{n+1} a_i - \prod_{i=1}^{n+1} b_i\right| &= \left|a_{n+1} \prod_{k=1}^{n} a_i - b_{n+1} \prod_{i=1}^{n} b_i\right| \\ &\leq |a_{n+1} - b_{n+1}| \cdot \prod_{i=1}^{n} |a_i| + |b_{n+1}| \cdot \left|\prod_{k=1}^{n} a_i - \prod_{k=1}^{n} b_i\right| \\ &\leq |a_{n+1} - b_{n+1}| \cdot \prod_{i=1}^{n} |a_i| + \sum_{k=1}^{n} \left[\left(\prod_{i=1}^{k-1} |a_i|\right) |a_k - b_k| \left(\prod_{j=k+1}^{n} |b_j|\right) \cdot |b_{n+1}|\right] \\ &= \sum_{k=1}^{n+1} \left[\left(\prod_{i=1}^{k-1} |a_i|\right) |a_k - b_k| \left(\prod_{j=k+1}^{n+1} |b_j|\right)\right]. \end{aligned}$$

□

Lemma A.6 *$(1 - 2^{-x})^m \geq 1 - 2^{-x+m-1}$ for all $m \in \mathbb{N}$ and all real number $x > 0$.*

Proof. The proof proceeds by induction on m. In the base cases that $m \in \{0, 1\}$, the claim is trivial. So, we assume $m \geq 1$. The induction hypothesis says that $(1 - 2^{-x})^{m-1} \geq 1 - 2^{-x+m-2}$. Then, we have

$$(1 - 2^{-x})^m = (1 - 2^{-x})^{m-1}(1 - 2^{-x}) \geq (1 - 2^{-x+m-2})(1 - 2^{-x})$$

$$\begin{aligned} &= 1 - 2^{-x+m-2} - 2^{-x} + 2^{-2x+m-2} \quad \geq \quad 1 - 2 \cdot 2^{-x+m-2} \\ &= 1 - 2^{-x+m-1}. \end{aligned}$$

□

Lemma A. 7 *For any natural numbers k and n $(0 \leq k \leq n)$ and any real number $\epsilon \in (0, \frac{1}{2})$, let $s_k^n = \binom{n}{k}(\frac{1}{2}+\epsilon)^k(\frac{1}{2}-\epsilon)^{n-k}$. If $2(k+1) \leq n$, then $s_k^n < s_{k+1}^n$.*

Proof. Suppose $2(k+1) \leq n$. Notice that this assumption implies that $2k+1 < n$ since k and n are integers.

$$\begin{aligned} s_{k+1}^n - s_k^n &= \frac{n!}{(k+1)!(n-k-1)!}\left(\frac{1}{2}+\epsilon\right)^{k+1}\left(\frac{1}{2}-\epsilon\right)^{n-k-1} - \frac{n!}{k!(n-k)!}\left(\frac{1}{2}+\epsilon\right)^{k}\left(\frac{1}{2}-\epsilon\right)^{n-k} \\ &= \frac{n!}{(k+1)!(n-k-1)!}\left(\frac{1}{2}+\epsilon\right)^{k}\left(\frac{1}{2}-\epsilon\right)^{n-k-1}\left[(n-k)\left(\frac{1}{2}+\epsilon\right) - (k+1)\left(\frac{1}{2}-\epsilon\right)\right]. \end{aligned}$$

Write L for $\frac{n!}{(k+1)!(n-k-1)!}\left(\frac{1}{2}+\epsilon\right)^k\left(\frac{1}{2}-\epsilon\right)^{n-k-1}$. Then, it follows that

$$s_{k+1}^n - s_k^n = L \cdot \left[\frac{1}{2}(n-2k-1) + (n+1)\epsilon\right] \geq L(n+1)\epsilon > 0.$$

Therefore, we conclude that $s_{k+1}^n > s_k^n$. □

Below we shall state Stirling's formula without proofs.

Lemma A. 8 **(Stirling's Formula)** *$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\frac{1}{6n}+h(n))$, where $h \in O(\frac{1}{n^2})$ and e is the base of the natural logarithm.*

Lemma A. 9 *For any sufficiently large natural number n, $\binom{n}{\lfloor n/2 \rfloor} \leq \frac{2^n}{\sqrt{\pi n}}(1+\frac{1}{6n}) \leq \frac{2^{n+1}}{\sqrt{\pi n}}$.*

Proof. Suppose that n is even and is of the form $2m$. Using Stirling's formula, for any sufficiently large integer n,

$$\begin{aligned} \binom{n}{\lfloor n/2 \rfloor} &= \frac{(2m)!}{(m!)^2} \quad \leq \quad \frac{\sqrt{4\pi m}(\frac{2m}{e})^{2m}(1+\frac{1}{12m})}{2\pi m(\frac{m}{e})^{2m}} \\ &\leq \frac{2^{2m}}{\sqrt{2\pi m}}\left(1+\frac{1}{12m}\right) \quad = \quad \frac{2^n}{\sqrt{\pi n}}\left(1+\frac{1}{6n}\right) \\ &\leq \frac{2^{n+1}}{\sqrt{\pi n}} \end{aligned}$$

since $1+\frac{1}{12m} < 2$.

Next we suppose that $n = 2m+1$.

$$\binom{n}{\lfloor n/2 \rfloor} = \frac{(2m+1)!}{m!(m+1)!} \quad \leq \quad \frac{\sqrt{2\pi(2m+1)}(\frac{2m+1}{e})^{2m+1}(1+\frac{1}{12(2m+1)})}{\sqrt{2\pi m}(\frac{m}{e})^m \cdot \sqrt{2\pi(m+1)}(\frac{m+1}{e})^{m+1}}$$

$$\begin{aligned}
&\leq \frac{1}{e\sqrt{2\pi}} \cdot \sqrt{\frac{2m+1}{m(m+1)}} \cdot \frac{(2m+1)^{2m+1}}{m^m \cdot (m+1)^{m+1}} \cdot \left(1 + \frac{1}{12(2m+1)}\right) \\
&\leq \frac{2^{2m+1}}{e\sqrt{2\pi(2m+1)}} \cdot \left(1 + \frac{1}{2m}\right) \cdot \left(1 + \frac{1}{2m}\right)^m \cdot \left(1 + \frac{1}{12(2m+1)}\right) \\
&\leq \frac{2^{2m+1}}{\sqrt{2e\pi(2m+1)}} \cdot \left(1 + \frac{1}{2m}\right)^m \cdot \left(1 + \frac{1}{12(2m+1)}\right).
\end{aligned}$$

The last inequality follows from the fact that $1 + \frac{1}{2m} < \sqrt{e}$. From the fact that $(1 + \frac{1}{2m})^m \leq \sqrt{e}$, it follows that

$$\binom{n}{\lfloor n/2 \rfloor} \leq \frac{2^{2m+1}}{\sqrt{2e\pi(2m+1)}} \cdot \sqrt{e} \cdot \left(1 + \frac{1}{12(2m+1)}\right) \leq \frac{2^n}{\sqrt{2\pi n}} \cdot \left(1 + \frac{1}{12n}\right) \leq \frac{2^{n+1}}{\sqrt{2\pi n}}.$$

□

Lemma A.10 *Let I be a finite index set. Let $\{\mathcal{E}_i\}_{i \in I}$ be a partition of the sample space Ω. Then, for any event $\mathcal{E}$,*

$$\mathbf{Pr}[\mathcal{E}] = \sum_{i \in I} \mathbf{Pr}[\mathcal{E} \mid \mathcal{E}_i] \cdot \mathbf{Pr}[\mathcal{E}_i].$$

Proof. For simplicity, assume that $I = \{1, 2, \ldots, k\}$. By the definition of the conditional probability, $\mathbf{Pr}[\mathcal{E} \mid \mathcal{E}_i] = \mathbf{Pr}[\mathcal{E} \cap \mathcal{E}_i]/\mathbf{Pr}_{\mathcal{E}_i}[\,]$ if $\mathbf{Pr}[\mathcal{E}_i] > 0$. Hence, we get $\mathbf{Pr}[\mathcal{E} \mid \mathcal{E}_i] \cdot \mathbf{Pr}[\mathcal{E}_i] = \mathbf{Pr}[\mathcal{E} \cap \mathcal{E}_i]$.

Since $\mathcal{E}_i$ and $\mathcal{E}_j$ are disjoint if $i \neq j$, the sum of all $\mathbf{Pr}[\mathcal{E} \mid \mathcal{E}_i] \cdot \mathbf{Pr}[\mathcal{E}_i]$ is calculated as follows:

$$\begin{aligned}
\sum_{i=1}^{k} \mathbf{Pr}[\mathcal{E} \mid \mathcal{E}_i] \cdot \mathbf{Pr}[\mathcal{E}_i] &= \sum_{i=1}^{k} \mathbf{Pr}[\mathcal{E} \cap \mathcal{E}_i] = \mathbf{Pr}[\bigcup_{i=1}^{k} (\mathcal{E} \cap \mathcal{E}_i)] \\
&= \mathbf{Pr}[\mathcal{E} \cap (\bigcup_{i=1}^{k} \mathcal{E}_i)] = \mathbf{Pr}[\mathcal{E}].
\end{aligned}$$

□

Given a series $\{A_i\}_{i \in \mathbb{N}}$ of sets, the notation $\limsup_n A_n$ denotes the set $\bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} A_k$ and is called the *limits superior of* $\{A_i\}_{i \in \mathbb{N}}$. Note that $w \in \limsup_n A_n$ if and only if w lies in infinitely many of the A_n.

Lemma A.11 **(Borel-Cantelli Lemma)** *Let $\{A_i\}_{i \in \mathbb{N}}$ be a sequence of events.*

1. *If $\sum_{n=0}^{\infty} \mathbf{Pr}[A_n]$ converges, then $\mathbf{Pr}[\limsup_n A_n] = 0$.*
2. *Assume that $\{A_n\}_{n \in \mathbb{N}}$ is independent. If $\sum_{n=0}^{\infty} \mathbf{Pr}[A_n]$ diverges, then $\mathbf{Pr}[\limsup_n A_n] = 1$.*

Proof. (1) Assume that $\sum_{n=0}^{\infty} \mathbf{Pr}[A_n]$ converges. Let k be any natural number. From the fact that $\limsup_n A_n \subseteq \bigcup_{i=k}^{\infty} A_i$, it follows that

$$\mathbf{Pr}[\limsup_n A_n] \leq \Pr\left[\bigcup_{i=k}^{\infty} A_i\right] \leq \sum_{i=k}^{\infty} \mathbf{Pr}[A_i].$$

By our assumption, $\lim_{i\to\infty}\sum_{i=k}^{\infty}\mathbf{Pr}[A_i]=0$. Therefore, $\mathbf{Pr}[\limsup_n A_n]=0$.

(2) Assume that $\{A_i\}_{i\in\mathbb{N}}$ is an independent sequence of events. Assume also that $\sum_{i=0}^{\infty}\mathbf{Pr}[A_i]$ diverges. It suffices to prove that $\mathbf{Pr}[\bigcup_{n=0}^{\infty}\bigcap_{k=n}^{\infty}A_k^c]=0$, where A_k^c is the complement of A_k (i.e., $\Omega-A_k$). For this claim, we want to show that $\mathbf{Pr}[\bigcap_{k=n}^{\infty}A_k^c]=0$ for all $n\in\mathbb{N}$.

Note that $\mathbf{Pr}[\bigcap_{k=n}^{\infty}A_k^c]=\lim_{j\to\infty}\mathbf{Pr}[\bigcap_{k=n}^{n+j}A_k^c]$. Let e be the base of the natural logarithm. Since $1-z\le e^{-z}$ holds,

$$\mathbf{Pr}\left[\bigcap_{k=n}^{n+j}A_k^c\right]=\prod_{k=n}^{n+j}\mathbf{Pr}[A_k^c]=\prod_{k=n}^{n+j}n+j(1-\mathbf{Pr}[A_k])\le\exp\left[-\sum_{k=n}^{n+j}\mathbf{Pr}[A_k]\right],$$

where $\exp[z]$ means e^z. Let us consider the last expression and denote it by T_n^j. Since $\sum_{i=0}^{\infty}\mathbf{Pr}[A_i]$ diverges, the value T_n^j tends to 0 as j approaches infinity. Thus, $\lim_{j\to\infty}\mathbf{Pr}[\bigcap_{k=n}^{n+j}A_k^c]=0$. □

Lemma A.12 **(Hölder's Inequality)** *Let I be a finite index set. For any two sets of positive real numbers $\{a_i\}_{i\in I}$ and $\{b_i\}_{i\in I}$,*

$$\sum_{i\in I}a_i\cdot b_i\le\left(\sum_{i\in I}a_i\right)^{1/p}\cdot\left(\sum_{i\in I}b_i\right)^{1/q},$$

where $\frac{1}{p}+\frac{1}{q}=1$.

Bibliography

[1] E.W. Allender and R. Rubinstein, P-printable sets, *SIAM J. Comput.*, **17** (1988), 1193–1202.

[2] D. Angluin, On counting problems and the polynomial hierarchy, *Theoretical Computer Science* **12** (1980), pp.161–173.

[3] T. Baker, G. Gill, and R. Solovay, Relativizations of the P=?NP question, *SIAM J. Comput.* **4** (1975), 431–442.

[4] J. L. Balcázar, J. Díaz and J. Gabarró, *Structural Complexity I, II*, Springer-Verlag, 1988(I), 1990(II).

[5] J.L Balcázar and U. Schöning, Bi-immune sets for complexity classes, *Mathematical Systems Theory*, **18** (1985), pp.1–10.

[6] J. Belanger and J. Wang, Rankable distributions do not provide harder instances than uniform distributions, *in* "Proceedings, 1st Computing and Combinatorics Conference," 1995, Lecture Notes in Computer Science, Vol.959, pp.410–419.

[7] J. Belanger and J. Wang, Reductions and convergence rates of average time, *in* "Proc. 2nd International Computing and Combinatorics Conference", Lecture Notes in Computer Science, Vol.1090, pp.300–309, 1996.

[8] C.H. Bennett and J. Gill, Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1, *SIAM J. Comput.* **10** (1981), 96–113.

[9] S. Ben-David, B. Chor, O. Goldreich, and M. Luby, On the theory of average case complexity, *J. Comput. System Sci.* **44** (1992), 193–219. A preliminary version appeared in Proceedings, 22nd STOC, 1990, pp.379–386.

[10] L. Berman and J. Hartmanis, On isomorphism and density of NP and other complete sets, *SIAM J. Comput.* **6** (1977), 305–322.

[11] P. Billingsley, *Probability and Measure*, John Wiley & Sons, New York, 1995.

[12] A. Blass and Y. Gurevich, Randomized reductions of search problems, *SIAM J. Comput.* **22** (1993), 949–975.

[13] B. Bollobas, T.I. Fenner and A.M. Frieze, An algorithm for finding Hamiltonian cycle in a random graph, *in* "Proceedings, 17th ACM Symposium on Theory of Computing", 430–439, 1985.

[14] R.V. Book, Tally languages and complexity classes, *Information and Control*, **26** (1974), pp.186–193.

[15] R.V. Book, Comparing complexity classes, *J. Comput. System Sci.* **9** (1974), 213–229.

[16] R. Book and D.Z. Du, The existence and density of generalized complexity cores, *J. of ACM*, **34** (1987), 718–730. 1987.

[17] R. Book, D.Z. Du, and D. Russo, On polynomial and generalized complexity cores, *in* "Proceedings, 3rd Conference on Structure in Complexity Theory", pp.236–250, 1988.

[18] J. Cai and L. A. Hemachandra, On the power of parity polynomial time, *Math. Systems Theory*, **23** (1990), pp.95–106.

[19] J. Cai and A. Selman, Fine separation of average time complexity classes, *in* "Proceedings, 13th Annual Symposium on Theoretical Computer Science", Lecture Notes in Computer Science, Vol.1046, pp.331–343, 1996.

[20] J. Carter and M. Wegman, Universal classes of hash functions, *J. Comput. System Sci.*, **18** (1979), pp.143–154.

[21] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer, Alternation, *J. ACM* **28** (1981), 114–133.

[22] S. Cook, The complexity of theorem proving procedures, *in* "Proceedings, 3rd ACM Symposium on Theory of Computing, pp.151–158, 1971.

[23] K. de Leeuw, E.F. Moore, C.E. Shannon, and N. Shapiro, Computability by probabilistic machines, *in* C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pp.183–212, Princeton University Press, Princeton, 1955.

[24] D.Z. Du, T. Isakowitz, and D.A. Russo, Structural properties of complexity cores, unpublished manuscript, Department of Mathematics, University of California at Santa Barbara, 1984.

[25] J. Franco and M. Paull, Probabilistic analysis of the Davis Putnum procedure for solving the satisfiability problem, *Discrete Applied Math.* **5** (1983), 77–87.

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

[27] J. Geske, D. Huynh, and J. Seiferas, A note on almost-everywhere complex sets with application to polynomial complexity degrees, *Information and Computation* **92** (1991), pp.97–104.

[28] J. Gill, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6** (1977), 675–695.

[29] M. Goldmann, P. Grape, and J.Håstad, On average time hierarchy, *Inform. Processing Letters* **49** (1994), 15–20.

[30] O. Goldreich, Towards a theory of average case complexity (a survey), Technical Report No.507, Israel Institute of Technology, 1988.

[31] J. Goldsmith, L.A. Hemachandra, D. Joseph, and P. Young, Near-testable sets, *SIAM. J. Comput.* **20** (1991), pp.506–523.

[32] P. Grape, Two results in average case complexity, Technical Report TRITA-NA-9105, Royal Institute of Technology, Stockholm, Sweden, 1991.

[33] J. Grollman and A.L. Selman, Complexity measures for public cryptosystems, *SIAM J. Comput.* **17** (1988), 309–335.

[34] Y. Gurevich, Complete and incomplete randomized NP problems, *in* "Proceedings, 28th IEEE Symposium on Foundations of Computer Science", pp.111–117, 1987.

[35] Y. Gurevich, Matrix decomposition is complete for the average case, *in* "Proceedings, 31st IEEE Symposium on Foundations of Computer Science", pp.802–811, 1990.

[36] Y. Gurevich, Average case completeness, *J. Comput. System Sci.* **42** (1991), pp.346–398.

[37] Y. Gurevich and S. Shelah, Expected computation time for Hamiltonian path problem, *SIAM J. Comput.* **16** (1987), pp.486–502.

[38] J. Hartmanis, Generalized Kolmogorov complexity and the structure of feasible computations, *in* "Proceedings, 24th IEEE Symposium on the Foundations of Computer Science," pp.439–445, 1983.

[39] J. Hartmanis, N. Immerman, and V. Sewlson, Sparse sets in NP−P: EXPTIME versus NEXPTIME, *Information and Control* **65** (1985), pp.159–181.

[40] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, Construction of a pseudo-random generator from any one-way function, Technical Report, TR-91-068, International Computer Science Institute, Berkeley, California, 1991. Preliminary versions appeared in the proceedings of the 21st STOC, 1989, pp.12–24 and the 22nd STOC, 1990, pp.395–404.

[41] M. Hermo and E. Mayordomo, A note on polynomial-size circuits with low resource-bounded, *Math. Systems Theory, 27* (1994), pp.347–356.

[42] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Massachusetts, 1979.

[43] R. Impagliazzo, A personal view of average-case complexity, *in* "Proceedings, 10th Annual Structure in Complexity Theory Conference," pp.134–147, 1995.

[44] R. Impagliazzo and L.A. Levin, No better ways to generate hard NP-instances than picking uniformly at random, *in* "Proceedings, 31th IEEE Symposium on Foundations of Computer Science", pp.812–821, 1990.

[45] D.S. Johnson, A catalog of complexity classes, Chapter 2 of *Handbook of Theoretical Computer Science*, edited by J. van Leeuwen, Elsevier Science Publishers B.V., 1990.

[46] D. Joseph and P. Young, Some remarks on witness functions for nonpolynomial and noncomplete sets in NP, *Theoret. Comput. Sci.* **39** (1985), 225–237.

[47] J. Kadin, The polynomial hierarchy collapses if the Boolean hierarchy collapses, *SIAM J. Comput.* **17** (1988), 1263–1283. Its erratum appeared in: *SIAM J. Comput.* **20** (1991), p.404.

[48] C. Karg and R. Schuler, Structure in average case complexity, *in* "Proceedings, 6th Annual International Symposium on Algorithms and Computations," Lecture Notes in Computer Science, Vol.1004, pp.62–71, 1995.

[49] R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations* (R. Miller and J. Thatcher, eds.), pp.85–104, Plenum Press, new York, 1972.

[50] R.M. Karp and R.J. Lipton, Turing machines that take advice, *Enseign. Math.*, **28** (1982), pp.191–209.

[51] P.M.W. Knijnenburg, On randomizing decision problems: a survey of the theory of randomized NP, Technical Report RUU-CS-88-15, Department of Computer Science, University of Utrecht, 1988.

[52] K. Ko, Some observations on the probabilistic algorithms and NP-hard problems, *Information Processing Letters*, **14** (1982), pp.39–43.

[53] K. Ko, On some natural complete operations, *Theoretical Computer Science*, **37** (1985), pp.1–30.

[54] K. Ko, *Complexity Theory of Real Functions*, Birkhäuser, Boston, 1991.

[55] K. Ko and H. Friedman, Computational complexity of real functions, *Theoretical Computer Science*, **20** (1982), pp.323–352.

[56] K.I. Ko and D. Moore, Completeness, approximation and density, *SIAM J. Comput.*, **10** (1981), pp.787–796.

[57] S.A. Kurtz, On the random oracle hypothesis, *Information and Control* **57** (1983), pp.40–47.

[58] C. Lautemann, BPP and the polynomial time hierarchy, *Information Processing Letters*, **17** (1983), pp.215–218.

[59] L.A. Levin, Universal sequential search problems, *Problems of Information Transmission* **9** (1973), pp.265–266.

[60] L. Levin, Average case complete problems, *SIAM J. Comput.* **15** (1986), 285–286. A preliminary version appeared under the title "Problems, complete in "average" instance" *in* "Proceedings, 16th ACM Symposium on Theory of Computing", p.465, 1984.

[61] M. Li and P. M.B. Vitányi, Average case complexity under the universal distribution equals worst-case complexity, *Information Processing Letters*, **42** (1992), pp.145–149.

[62] M. Li and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*, Springer-Verlag, New York, 1993.

[63] J.H. Lutz, Category and measure in complexity classes, *SIAM J. Comput.* **19** (1990), 1100–1131.

[64] J.H. Lutz, Almost everywhere high nonuniform complexity, *J. of Comput. System Sci.* **44** (1992), 220–258.

[65] J. Machta and R. Greenlaw, The computational complexity of generating random fractals, Technical Report, TR 93-04, University of Massachusetts at Amherst, 1993.

[66] S.R. Mahaney, Sparse complete sets for NP: solution of a conjecture by Berman and Hartmanis, *J. Comput. System Sci.* **25** (1988), 130–143.

[67] J.A. Makowsky and A. Sharell, On average case complexity of SAT for symmetric distribution, *J. Logic Computat.* (1995), 71–92.

[68] Y. Mansour, N. Nisan, and P. Tiwari, The computational complexity of universal hashing, *Theoret. Comput. Sci.* **107** (1993), 121–133.

[69] E. Mayordomo, Almost every set in exponential time is P-bi-immune, *Theoret. Comput. Sci.* **136** (1994), 487–506.

[70] E. Mayordomo, Contributions to the study of resource-bounded measure Ph.D. dissertation, Sistemes Informàtics, Universitat Politècnica de Catalunya, 1994.

[71] A. Meyer and M. Paterson, With what frequency are apparently intractable problems difficult ?, Technical Report MIT/LCS/TM-126, MIT, 1979.

[72] A.R. Meyer and L.J. Stockmeyer, The equivalence problem for regular expressions with squaring requires exponential time, *in* "Proceedings, 13th IEEE Symposium on Switching and Automata Theory", 125–129, 1972.

[73] B. Meyer, Constructive separation of classes of indistinguishable ensembles, "Proceedings, 9th IEEE Annual Conference on Structure in Complexity Theory", pp.198–204, 1994.

[74] P.B. Milterson, The complexity of malign ensembles, *SIAM J. Comput.* **22** (1993), 147–156.

[75] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, 1995.

[76] M. Mundhenk and R. Schuler, Random languages for non–uniform complexity classes, *Journal of Complexity* **7** (1991), 296–310.

[77] P. Orponen, D.A. Russo, and U. Schöning, Optimal approximations and polynomially levelable sets, *SIAM J. Comput.* **15** (1986), 399–408.

[78] P. Orponen and U. Schöning, On the density and complexity of polynomial cores for intractable sets, *Inform. Control*, **70** (1986), pp.54–68.

[79] R. Ostrovsky and A. Wigderson, One-way functions are essential for non-trivial zero-knowledge, *in* "Proceedings, 2nd Israel Symposium on Theory of Computing and Systems," Israel, 1993. An extended abstract appeared as Technical report, ICSI-TR-93-073, International Computer Science Institute at Berkeley, 1993

[80] C.H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994.

[81] C. Papadimitriou and S. Zachos, Two remarks on the power of counting, *in* "Proceedings, 6th GI Conference on Theoretical Computer Science," Lecture Notes in Computer Science, Vol.145, pp.269–275, Springer-Verlag, Berlin, 1983.

[82] W.J. Paul, N. Pippenger, E. Szemerédi, and W.T. Trotter, On determinism versus nondeterminism and related problems, *in* "Proceedings, 24th IEEE Symposium on Foundations of Computer Science", pp.429–438, 1983.

[83] A. Pavan and A.L. Selman, Complete distributional problems, hard languages, and resource-bounded measure, manuscript, Department of Computer Science, State University of New York at Buffalo, 1996.

[84] R. Reischuk and C. Schindelhauer, Precise average case complexity, *in* "Proceedings, 10th Symposium on Theoretical Aspect of Computer Science", Lecture Notes in Computer Science, Vol.665, Springer-Verlag, 650–661, 1993.

[85] K.W. Regan, Minimum-complexity pairing functions, *J. Comput. System Sci.* **45** (1992), 285–295.

[86] E.S. Santos, Probabilistic Turing machines and Computability, *Proc. Amer. Math. Soc.* **22** (1969), pp.704–710.

[87] H. Satoh and O. Watanabe, On set size comparison problem, Technical Report of IEICE, 1993.

[88] R.E. Schapire, The emerging theory of average-case complexity, Technical Report, MIT/LCS/TM-431, Massachusetts Institute of Technology, 1990.

[89] U. Schöning, A low and a high hierarchy within NP, *J. Comput. System Sci.* **27** (1983), 14–28.

[90] U. Schöning, Robust algorithms: a different approach to oracles, *Theoretical Computer Science* **40** (1985), pp.57–60.

[91] U. Schöning, *Complexity and structure*, Lecture Notes in Computer Science, Vol.211, Springer-Verlag, 1986.

[92] R. Schuler, Some properties of sets tractable under every polynomial-time computable distribution, *Information Processing Letters*, **55** (1995), pp.179–184.

[93] R. Schuler, Average polynomial time is hard for exponential time under sn-reductions, *in* "Proceedings, 15th Foundations of Software Technology and Theoretical Computer Science," Lecture Notes in Computer Science, Vol.1026, pp.240–247, 1995.

[94] R. Schuler, Truth-table closure and Turing closure of average polynomial time have different measure in **EXP**, *in* "Proceedings, 11th Annual Conference on Computational Complexity," pp.190–197, 1996.

[95] R. Schuler, A note on universal distributions for polynomial-time computable distributions, *in* "Proceedings, 12th Annual IEEE Conference on Computational Complexity," 1997.

[96] R. Schuler and O. Watanabe, Towards average-case complexity analysis of NP optimization problems, *in* "Proceedings, 10th Annual Conference on Structure in Complexity Theory," pp.148–159, 1995.

[97] R. Schuler and T. Yamakami, Structural average case complexity, *J. Comput. System Sci.* **52** (1996), 308–327. A preliminary version appeared *in* "Proceedings, 12th Foundations of Software Technology and Theoretical Computer Science," Lecture Notes in Computer Science, Vol.652, pp.128–139, Springer-Verlag, Berlin, 1992.

[98] R. Schuler and T. Yamakami, Sets computable in polynomial time on average, *in* "Proceedings of the 1st Computing and Combinatorics Conference," Lecture Notes in Computer Science, Vol.959, pp.400-409, 1995, Springer-Verlag.

[99] J.I. Seiferas, M.J. Fischer, and A.R. Meyer, Separating nondeterministic time complexity classes, *J. ACM*, **25** (1978), pp.146–167.

[100] M. Sipser, A complexity theoretic approach to randomness, *in* "Proceedings, 15th ACM Symposium on the Theory of Computing," pp.330–335, 1983.

[101] L.J. Stockmeyer, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977), 1–22.

[102] S. Toda, PP is as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **20** (1991), pp.865–877.

[103] L.G. Valiant, Relative complexity of checking and evaluating, *Information processing Letters* **5** (1976), pp.20–23.

[104] L. Valiant, The complexity of computing the permanent, *Theoretical Computer Science*, **5** (1979), 189–201.

[105] L.G. Valiant and V.V. Vazirani, NP is as easy as detecting unique solutions, *Theoretical Computer Science* **47** (1986), pp.85–93.

[106] R. Venkatesan and L. Levin, Random instances of a graph coloring problem are hard, *in* "Proceedings, 20th ACM Symposium on Theory of Computing", pp.217–222, 1988.

[107] R. Venkatesan and S. Rajagopalan, Average case intractability of Diophantine and matrix problems, *in* "Proceedings, 24th ACM Symposium on Theory of Computing", pp.632–642, 1992.

[108] J. Wang, Average case completeness of a word problem for groups, *in* "Proceedings, 27th ACM Symposium on Theory of Computing", pp.325–334, 1995.

[109] J. Wang, Average-case computational complexity theory, in *Complexity Theory Retrospective II*, editors, A. Selman and L. Hemaspaandra, Springer-Verlag, 1996.

[110] J. Wang, Average-case intractable NP problems, manuscript, 1996.

[111] J. Wang and J. Belanger, On average P vs. average NP, in *Complexity Theory – Current Research*, editors K. Ambos-Spies, S. Homer and U. Schöning, Cambridge University press, 47–67, 1993. A preliminary version appeared *in* "Proceedings, 7th IEEE Conference on Structure in Complexity Theory," pp.318–326, 1992.

[112] J. Wang and J. Belanger, On the NP-isomorphism problem with respect to random instances, *J. Comput. System Sci.*, **50** (1995), 151–164. A preliminary version appeared in "Proceedings, 8th Conference on Structure in Complexity Theory," 1993, pp.65–74 under the title: Isomorphisms of NP complete problems on random instances.

[113] O. Watanabe, Test instance generation for promised NP search problems, *in* "Proceedings, 9th Annual Conference on Structure in Complexity Theory," pp.205–216, 1994.

[114] O. Watanabe, personal communication, 1996.

[115] R.L. Wheeden and A. Zygmund, *Measure and Integral*, Marcel Dekker, New York, 1977.

[116] R.E. Wilber, Randomness and the density of hard problems, *in* "Proceedings, 24th IEEE Symposium on Foundations of Computer Science", pp.335–342, 1983.

[117] H. Wilf, Some examples of combinatorial averaging, *American Math. Monthly* **92** (1985), 250–261.

[118] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1977), 23–33.

[119] T. Yamakami, Polynomial time samplable distributions, *in* "Proceedings, 21st International Symposium on Mathematical Foundation of Computer Science," Lecture Notes in Computer Science, Vol.1113, pp.566–578, 1996.

[120] A.C. Yao, Theory and applications of trapdoor functions, *in* "Proceedings, 23rd Annual IEEE Symposium on Foundations of Computer Science," 1982, pp.80–91.

[121] C. Yap, Some consequences of non-uniform conditions on uniform classes, *Theoret. Comput. Sci.* **26** (1983), pp.287–300.

[122] S. Zachos, Collapsing probabilistic polynomial hierarchies, *in* "Proceedings, Conference on Computational Complexity Theory," Santa Barbara, 1983, pp.75–81.

[123] S. Zachos, Probabilistic quantifiers and games, *J. Comput. System Sci.* **36** (1988), pp.433–451.

List of Notation

Index

B

C

I

L

M

N

S

T

U

V

W