Average-Case Complexity Theory and Polynomial-Time Reductions

by
A. Pavan

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF STATE UNIVERSITY OF NEW YORK AT BUFFALO
IN PARTIAL FULFILLMENTS OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
August, 2001

Foreword

Several of the results in this thesis have appeared as refereed publications [13, 25, 67, 69]. The results in Section 4.1 were published [13] with J. Belanger and J. Wang as coauthors, and these results improved earlier results of Belanger and Wang [14]. The results in Section 5.2 were obtained jointly with L. Fortnow and A. Selman, and these results were first presented at the 6th Annual International Computing and Combinatorics Conference [24]. The results in Chapter 7 were first announced at the 16th Annual IEEE Conference on Computational Complexity [68].

Acknowledgments

I take this opportunity to express my deep sense of gratitude to Prof. Alan Selman, my thesis advisor, for providing me guidance in all aspects of this work. His immense knowledge, his deep insights into the field, and his ability to pose the correct question always filled me with admiration. I am very grateful to Alan for his support and encouragement during my unproductive phases. I have learnt a great deal about the art of technical paper writing and presentation skills from him. I also thank Alan for his advice on various non-technical matters.

I am grateful to Prof. Jin-Yi Cai from whom I have learnt a lot. Jin-Yi's probing questions during my talks at the Complexity Theory Research Group meetings helped me understand the concepts better. I also learnt what it means by "understanding a technical paper" from him. I thank Jin-Yi for his help even after he moved to Madison.

I thank Prof. Kenneth Regan for all the help over the years. Ken always encouraged my work, I also thank Ken for his comments on my thesis. Ken gave me ride for numerous theory meetings at Rochester and to complexity conferences.

I am thankful to Prof. Lane Hemaspaandra for agreeing to be my outside reader and for his constructive suggestions which have led to an improved presentation of this thesis.

I am extremely grateful to Prof. Somenath Biswas who introduced me to the beautiful world of complexity theory. If it were not for his excellent courses at I. I. T., Kanpur, I would not have pursued Ph.D in complexity theory and would have missed all the excitement of the field.

I thank Prof. Lance Fortnow and Prof. Osamu Watanabe for their comments and the useful discussions I had with them. I am grateful to Prof. Jack Lutz who showed a lot of interest in my work and for his help. I owe a lot to Dr. D. Sivakumar for his guidance, advice, help, and encouragement throughout my graduate studies.

Special thanks to fellow theory students Ajay, Charles, and Samik. I will always remember our fun filled discussions and numerous attempts to prove nontrivial theorems.

I thank the Computer Science and Engineering Department for the opportunities they have provided. I thank the department and Alan for the financial support I have received. I am grateful to all the secretaries, especially Ellie, Sally, Gloria, and Joann, without whom nothing in the department would run. I thank Helene Kershner for giving me many opportunities to teach. Student life in the department has been really a lot of fun. I thoroughly enjoyed the Fall/Spring picnics, the grad conference, and the CSEGSA meetings. Thanks to all the people who made these possible.

It has been a pleasure living in Buffalo. I always enjoyed the record snow falls and numerous trips to Niagara Falls. I express my sincere thanks to all my friends at Buffalo, especially Almu, Anurag, Charles, Girish, John, Pallavi, Samik, Sanghamitra, Srikanth, and Vishal for the delightful company they have given. Many many thanks to Pallavi, to whom I owe a lot, for many many things. I thank all my squash partners for the wonderful times I had on the squash courts.

I thank all my long-time friends for their help. Special thanks to B. N. Srinivas who, during my Bachelors, got me interested in theory, and then several years later got me interested in jogging.

I am greatly indebted to my yoga teacher Sree L. Narasimhulu without whose help I would never have finished this thesis. I sincerely thank my high school teacher Sree John Solomon for his encouragement throughout my studies.

Finally, I am extremely grateful to my parents and my brother for the ever increasing love and encouragement they have shown. My interest in higher studies would never have been there, had my mother not taken special care in my studies during my childhood days. I am fortunate to have a great set of relatives - grand parents, aunts, uncles, and wonderful cousins.

It is with great pleasure I dedicate this thesis to all the people I mentioned here.

Contents

Abstract

This thesis studies average-case complexity theory and polynomial-time reducibilities. The issues in average-case complexity arise primarily from Cai and Selman's extension of Levin's definition of average polynomial time. We study polynomial-time reductions between distributional problems. Under strong but reasonable hypotheses we separate ordinary NP-completeness notions.

The average-case complexity of a problem is, in many cases, a more significant measure than the worst-case complexity. It is known that some NP-complete problems, such as k-coloring and Hamiltonian cycle, can be solved quickly on average. This has motivated the development of the study of average-case analysis of algorithms. Levin [56] initiated a general study of average-case complexity by defining a robust notion of average polynomial time and the notion of distributional NP-completeness. Cai and Selman [19] gave a general definition of T *on average* for arbitrary time bounds T. They observed difficulties with Levin's definition of average polynomial time when applied to unreasonable input distributions. Their definition of average polynomial time slightly modifies Levin's definition to avoid these difficulties. In this thesis we study reasonable distributions, distributionally-hard languages, and reductions among distributional problems.

We prove several results demonstrating that it suffices to restrict one's attention to reasonable distributions. This is important because both Levin's definition and Cai and Selman's definition yield unsatisfactory consequences when we permit unreasonable distributions. We show that if NP has a DTIME(2^n)-bi-immune language, then every DistNP-complete problem must have a reasonable distribution. We prove that the class $\mathrm{P}_{\mathrm{pcomp}}$, the set of languages that can be decided in average polynomial time with respect to every polynomial-time computable distribution, remains unchanged when restricted to reasonable distributions. We strengthen and present a simpler proof of a result of Belanger and Wang [14], which shows that Cai and Selman's definition of average-polynomial time is not closed under many-one reductions.

Cai and Selman showed that every is P-bi-immune language is distributionally-hard. We study the question of whether there exist distributionally-hard languages that are not P-bi-immune. First we show that such languages exist if and only if P contains P-printable-immune sets. Then we extend this

characterization significantly to include assertions about several traditional questions about immunity, about finding witnesses for NP-machines, and about the existence of one-way functions.

Next we study polynomial-time reducibilities. Ladner, Lynch, and Selman [54] showed, in the context of worst-case complexity, that various polynomial-time reductions differ in E. We show similar results for reductions between distributional problems and we show that most of the completeness notions for DistEXP are different.

Finally, we turn our attention to the question of whether various notions of NP-completeness are different. Lutz and Mayordomo, under the hypothesis that the p-measure of NP is not zero, and Ambos-Spies and Bentzien, under the hypothesis that NP has a p-generic language, showed that various completeness notions in NP are different. We introduce a structural hypothesis not involving stochastic properties and prove that the existence of a Turing complete language for NP that is not truth-table complete follows from our hypothesis. This result is the first to provide evidence that truth-table completeness for NP is weaker than Turing completeness for NP.

Chapter 1

Introduction

Alan Turing [83] described the *Turing machine*, a conceptual model of computation, and proposed that every "algorithm" can be programmed on a Turing machine. Around the same time, Church, Kleene, and Post developed formalizations, the λ-calculus, partial recursive functions, and Post machines, to capture the intuitive notion of "effectively computable." All these formalizations are provably equivalent to the Turing machine. Thus, they were able to identify "effective computability," a heretofore imprecise notion, with specific mathematical formalizations. Based on these formulations, computability became an object of formal study. Much of the early research in computability theory was dedicated to the study of decidability and the classification of undecidable problems into different degrees of unsolvability. It was not until the 1960s that researchers began a quantitative study of the computational resources, such as time and space, needed to solve decidable problems [35, 36]. This research has led to the classification of problems into various *complexity classes*. A complexity class is a collection of problems that can be solved using same amount of resources.

Commonly one identifies problems that can be solved in polynomial time (that belong to the complexity class P) as those that are "feasibly computable," and conversely, one identifies "intractability" with problems that cannot be solved in polynomial time [20]. For the purpose of distinguishing feasibility from intractability, the Turing machine remains a useful model of computation. There are at least two reasons for this. The first is that a Turing machine can simulate any other computational device with at most a polynomial slowdown. The second is that it is a simple model. As a consequence of the work of Cook [21], Karp [50], and Levin [55] in the early

1970's, we have the concept of NP-completeness and we know hundreds of NP-complete combinatorial problems. By definition a language L is NP-complete if L belongs to NP and every language in NP is polynomial-time many-one reducible to L. Because of this property, it is natural to think NP-complete problems as the "hardest problems" in NP. Since P is closed under polynomial-time many-one reductions, the existence of a polynomial-time algorithm for any one of the NP-complete problems implies the existence of polynomial-time algorithms for every problem in NP.

It is possible that a problem might not belong to P, yet some algorithm solving the problem runs in polynomial-time on most inputs. NP-completeness is a worst-case concept and in practice hard instances arise comparatively rarely. Some NP-complete problems have been shown to be very easy on average. For example Wilf [91] discovered a fast on average algorithm for the k-coloring problem, as did Gurevich and Shelah [33] for the Hamiltonian problem, both under the commonly used distribution on graphs. Because of these examples, as well as ordinary computational practice, the average-case complexity of a problem seems to be a more significant measure than the worst-case complexity.

Levin [56] was the first to advocate a general study of average-case complexity. An average-case complexity class consists of pairs (L, μ) called *distributional problems*, where L is a language and μ is a distribution over the input domain. Given a distributional problem, it is an important issue to either find an expected polynomial-time algorithm that solves the problem or prove that such an algorithm does not exist. Levin [56] provided two central notions for studying this issue. One is analogous to the class P, and provides an *easiness* notion; the other is analogous to the class of NP-complete sets, and provides a *hardness* notion. For the first, Levin defined a robust notion of what it means for the running time of an algorithm to be polynomial on average. Using this notion, Average-P denotes the set of all distributional problems (L, μ) such that there exists an algorithm for L whose running time is polynomial on μ-average. Let DistNP denote the collection of all distributional problems (L, μ) such that μ is computable in polynomial time and L belongs to NP. For the second notion, that of hardness, Levin defined reductions between distributional problems. Using these reducibilities, in the usual manner, he defined a distributional problem (L, μ) to be complete for DistNP if (L, μ) belongs to DistNP and every distributional problem in DistNP is reducible to (L, μ). Levin showed that distributional tiling with a simple distribution is complete for DistNP, and since then, several additional

DistNP-complete problems have been found [16, 32, 85, 86, 87, 89].

Levin's original ideas have been studied and enhanced by a number of researchers [15, 19, 32]. Levin's definition only distinguishes polynomial on average from super-polynomial on average. Ben-David *et al.* [15] proposed a straightforward generalization and gave a definition of T on average for an arbitrary time bound T. However, Cai and Selman [19] discovered that the definition of Ben-David *et al.* has some limitations. Let ATIME($T(n)$) denote the class of distributional problems that are decidable by a Turing machine whose running time is $T(n)$ on μ-average according to the definition of Ben-David *et al.* Cai and Selman observed that ATIME(c^n) = ATIME(2^n) for every positive constant c. One of the salient features of worst-case complexity theory is the existence of fine hierarchies. That is, given time bounds $T(n)$ and $T'(n)$, where $T'(n)$ grows sufficiently faster than $T(n)$, there are problems that are solvable in $T'(n)$ time but not in $T(n)$ time. The definition of Ben-David *et al.* fails to achieve a fine hierarchy. Cai and Selman also observed difficulties with Levin's definition when applied to distributions that decrease very quickly. For example there exists a language that does not belong to P almost everywhere, but still is in Average-P with respect to the distribution $\mu'(x) = 1/4^{|x|}$.

To avoid these difficulties, Cai and Selman [19] gave a new definition of *T on average* for an arbitrary time bound T. As a consequence of their definition, they obtained a hierarchy theorem for arbitrary average-case time bounds that is as tight as the hierarchy theorem of Hartmanis and Stearns [36] is for the worst case. Let AVP denote the class of distributional problems that can be solved in polynomial time according to Cai and Selman's definition. It follows from the definitions that AVP $\subseteq$ Average-P. Cai and Selman showed that the inclusion is proper but that the difference between Average-P and AVP is modest. Define a distribution to be *reasonable* if there exists a constant $s > 0$ such that $\mu(\{x \mid |x| \geq n\}) = \Omega(\frac{1}{n^s})$. Unreasonable distributions give too much weight to small instances and decrease too quickly. Cai and Selman showed that if μ is a reasonable distribution, then $(L, \mu) \in$ AVP if and only if $(L, \mu) \in$ Average-P. Also Cai and Selman introduced a new notion of hardness, called *distributional hardness*, in addition to the well-known worst-case hardness and almost-everywhere hardness. They define a language to be *distributionally-hard to recognize* if for every polynomial-time computable distribution μ, the distributional problem $(L, \mu) \notin$ AVP. They proved that almost-everywhere hardness implies distributional hardness i.e., every P-complex (equivalently P-bi-immune) language is distributionally-hard. It is easy to observe that distributional hardness implies worst-case hardness. We extensively study reasonable distributions and distributional hardness in this thesis.

Now we briefly describe the results in this thesis. Chapters 2 and 3 are preliminary chapters. Our results begin in Chapter 4. In Chapter 4 we present various results demonstrating that it is sufficient to restrict one's attention to reasonable distributions. This is important because criticisms of the all formulations of average-case complexity involve unreasonable distributions. We strengthen a result of Belanger and Wang [14] and give a simpler proof. We show that a class described by Schuler and Yamakami [76] remains unchanged when restricted to reasonable distributions. We show, under a reasonable assumption, that every DistNP-complete problem has a reasonable distribution.

In Chapter 5 we study the question of whether there exist languages that are distributionally-hard but not P-bi-immune. We prove that this question is equivalent to one first raised by Allender and Rubinstein [2]. Namely, we show that such languages exist if and only if P contains P-printable-immune sets. Then we significantly extend this characterization by proving that the question of the existence of distributionally-hard languages that are not P-bi-immune is equivalent to several previously studied hypotheses, including hypotheses about the existence of certain kinds of one-way functions, and about computing accepting computations of NP-machines.

The last two chapters concern polynomial-time reductions. Reductions enable us to compare the relative difficulty of problems. NP-completeness notion depends on reductions. Researchers have identified various types of reductions, such as *Turing* reductions, *many-one* reductions, *bounded truth-table* reductions, and *truth-table* reductions, in order to study relations among different problems. Ladner, Lynch, and Selman [54] were the first to compare these reductions. For each pair of reductions, they exhibited languages A and B, both in EXP, such that A is reducible to B via one of the reductions but not via the other reduction. Watanabe [90] showed that many-one completeness, bounded truth-table completeness, truth-table completeness, and Turing completeness for EXP are mutually different. Inspired by these results, in Chapter 6 we compare different notions of reductions among distributional problems. Here we show results similar to those of Ladner, Lynch, and Selman for reductions between distributional problems and we show that several completeness notions for DistEXP are different.

Ladner, Lynch, and Selman raised the natural questions of whether these reductions differ in NP and whether we can separate different NP-completeness notions. A positive answer to either would imply immediately that P $\neq$ NP. With this in mind, they conjectured that P $\neq$ NP implies that Turing reductions and many-one reductions differ on NP. However, this problem still remains open. So one of the goals is to separate these reductions under a reasonable hypothesis. Selman [78] obtained the first result that distinguishes reducibilities within NP. He showed that if NE$\cap$co-NE $\neq$ E, then Turing and many-one reductions are different in NP. For many years there was not much progress on separating completeness notions in NP. The first researchers to give technical evidence that Turing completeness and many-one completeness for NP differ were Lutz and Mayordomo [62], who in 1994 proved that if the *p-measure of NP* is not zero, then there exists a 3-truth-table complete set that is not many-one complete. Ambos-Spies and Bentzien [4] extended this result significantly. They used an hypothesis of resource-bounded category theory that is weaker than that of Lutz and Mayordomo to separate NP-completeness notions for nearly all of the bounded truth-table reducibilities.

It has been an open question whether we can separate NP-completeness notions without using hypotheses that involve essentially stochastic concepts. It has also remained open whether Turing completeness is different from truth-table completeness for NP. In this thesis we obtain results that answer these questions. In Chapter 7 we introduce a strong, but reasonable, structural hypothesis and prove that the existence of a Turing complete set for NP that is not truth-table complete follows from our hypothesis. This result is the first to provide evidence that truth-table completeness for NP is weaker than Turing completeness for NP.

Chapter 2

Preliminaries

In this chapter, we give basic definitions and review necessary concepts in complexity theory. We also give the notation that we use throughout the thesis.

2.1 Model of Computation and Languages

We fix the alphabet $\Sigma = \{0, 1\}$. Let Σ^* denote the set of all finite strings over the alphabet Σ. We denote the set of strings of length n, (less than or equal to n, greater than or equal to n) by Σ^n ($\Sigma^{\leq n}$, $\Sigma^{\geq n}$ respectively)[1] . A *language* is a subset of Σ^*. For $x \in \Sigma^*$ and $S \subseteq \Sigma^*$, $|x|$ denotes the length of x, and $||S||$ denotes the cardinality of S. Given two languages A and B, the *disjoint union* of A and B is the set,

$$A \oplus B = \{0x \mid x \in A\} \cup \{1y \mid y \in B\}.$$

We denote the *characteristic function* of a language A by $A(x) : A(x) = 1$ if $x \in A$, and $A(x) = 0$ otherwise. Let $\Sigma^* = \{z_n\}_n$, where z_n is the n-th string in the lexicographic order. Given a language A and a string x, $A|x = \{y \mid y < x \text{ and } y \in A\}$. We identify the initial segment $A|z_n$ with its characteristic sequence; i.e., $A|z_n = A(z_0) \cdots A(z_{n-1})$.

The *Turing machine* is our standard model of computation. We assume that the reader is familiar with deterministic and nondeterministic Turing

[1] Given a language L, it is common to denote the set of strings in L having length less than or equal to n by $L^{\leq n}$. Then $(\Sigma^*)^{\leq n}$ denotes the set of all strings having length less than or equal to n. For ease of notation, we replace this with $\Sigma^{\leq n}$.

machines. An *oracle Turing machine* is a multitape Turing machine with a special *query tape* and three distinguished states QUERY, YES, and NO. Should the machine enter state QUERY then the next state is YES or NO depending on whether the string currently written on the query tape belongs to the oracle. We denote an oracle Turing machine M with oracle A by M^A. We also consider *probabilistic Turing machines.* In addition to its usual tapes, a *probabilistic Turing machine* has a read-only one-way tape that contains an infinite binary sequence that can be regarded as outcomes of coin tosses. If the machine needs to make a random move, then it reads a bit from the random tape. A *transducer* is a Turing machine that computes functions. At the end of the computation the value of the function appears on a special write-only tape.

2.2 Complexity Classes

In this section we briefly review the definitions of several standard complexity classes. For detailed definitions, we refer the reader to standard text books [11, 45, 66]. Let P denote the class of languages that are accepted by polynomial-time-bounded deterministic Turing machines, and let NP denote the class of languages that are accepted by polynomial-time-bounded nondeterministic Turing machines. For every language L in NP, there exists a polynomial-time-computable relation R and a polynomial p such that for every x,

$$x \in L \Leftrightarrow \exists y \ \ [|y| \leq p(|x|) \text{ and } R(x, y)]. \tag{2.1}$$

A string y that satisfies the right hand side of (2.1) is called a *witness* of x.

A nondeterministic Turing machine M is *unambiguous* if for every string x, M has at most one accepting path on input x. A language is in UP if it is accepted by a polynomial-time-bounded unambiguous Turing machine.

Let DTIME$(T(n))$ denote the class of languages that are decided by $T(n)$ time-bounded deterministic Turing machines. NTIME$(T(n))$ is defined similarly using nondeterministic Turing machines. Hartmanis and Stearns [36] proved the following worst-case hierarchy theorem.

Theorem 2.1. ([36]) *Let $T(n)$ be a time-constructible function. Then there is a language L in* DTIME$(T(n))$ *such that for every function $T'(n)$, if $T'(n)\log(T'(n)) = o(T(n))$, then L is not in* DTIME$(T'(n))$.

Now we define several complexity complexity classes.

$$\begin{aligned}
\mathrm{P} &= \cup_{c=1}^{\infty}\mathrm{DTIME}(n^c + c)\\
\mathrm{NP} &= \cup_{c=1}^{\infty}\mathrm{NTIME}(n^c + c)\\
\mathrm{E} &= \cup_{c=1}^{\infty}\mathrm{DTIME}(2^{cn})\\
\mathrm{NE} &= \cup_{c=1}^{\infty}\mathrm{NTIME}(2^{cn})\\
\mathrm{EXP} &= \cup_{c=1}^{\infty}\mathrm{DTIME}(2^{n^c})\\
\mathrm{NEXP} &= \cup_{c=1}^{\infty}\mathrm{NTIME}(2^{n^c})
\end{aligned}$$

The *polynomial-time hierarchy*, PH, is defined by Stockmeyer [81] as

$$\begin{aligned}
\Sigma_0^P &= \mathrm{P}\\
\Sigma_k^P &= \mathrm{NP}^{\Sigma_{k-1}^P}, k \geq 1\\
\mathrm{PH} &= \cup_k \Sigma_k^P.
\end{aligned}$$

BPP is the class of all languages that are accepted by bounded-error polynomial-time-bounded probabilistic Turing machines. A language L belongs to BPP if there exists a polynomial-time-bounded probabilistic Turing machine M such that for every x,

$$\begin{aligned}
x \in L &\Rightarrow Pr[M \text{ accepts } x] > \frac{2}{3}, \text{ and}\\
x \notin L &\Rightarrow Pr[M \text{ accepts } x] < \frac{1}{3}.
\end{aligned}$$

RP is the class of languages that are accepted by one-sided error polynomial-time-bounded probabilistic Turing machines. A language L is in RP if there exists a polynomial-time bounded probabilistic Turing machine M such that for every x,

$$\begin{aligned}
x \in L &\Rightarrow Pr[M \text{ accepts } x] > \frac{2}{3}, \text{ and}\\
x \notin L &\Rightarrow Pr[M \text{ accepts } x] = 0.
\end{aligned}$$

Here the probabilities are taken over the coin tosses of the machines.

2.3 Reducibilities

Reductions translate instances of one problem to instances of another problem. We can compare relative hardness of languages using the notion of reductions.

A language A is *polynomial-time many-one reducible* to B, $A \le_m^p B$, if there exists a polynomial-time-computable function f such that for every x, $x \in A \Leftrightarrow f(x) \in B$. A is *Turing reducible* to B, $A \le_T^p B$, if A can be decided by a deterministic polynomial-time-bounded Turing machine with B as an oracle.

Ladner, Lynch, and Selman [54] defined various polynomial-time reducibilities in addition to the well-known Turing and many-one reducibilities. A language A is *truth-table* reducible to a language B, $A \le_{tt}^p B$, if there exist polynomial-time-computable functions g and h such that on input x, $g(x)$, for some $m \ge 0$, is (an encoding of) a set of queries $Q = \{q_1, q_2, \cdots, q_m\}$, and $x \in A$ if and only if $h(x, B(q_1), B(q_2), \cdots, B(q_m)) = 1$. The function g is the *truth-table generator* and h is the *truth-table evaluator*. For a constant $k > 0$, A is k*-truth-table* reducible to B, $A \le_{k\text{-}tt}^p B$, if for all x, $\|Q\| = k$, and A is *bounded-truth-table* reducible to B, $A \le_{btt}^p B$, if there is a constant $k > 0$ such that $A \le_{k\text{-}tt}^p B$. Selman [78] defined positive reductions. An oracle Turing machine M is *positive* if for each input x and oracle B, if M^B accepts x, then for all sets B' such that $B \subseteq B'$, $M^{B'}$ accepts x. A set A is *positive Turing* reducible to a set B, $A \le_{pos}^p B$, if A is Turing reducible to B via a positive oracle Turing machine. The following implications are obvious.

$$A \le_m^p B \Rightarrow A \le_{btt}^p B \Rightarrow A \le_{tt}^p B \Rightarrow A \le_T^p B.$$

$$A \le_m^p B \Rightarrow A \le_{pos}^p B \Rightarrow A \le_T^p B.$$

Ladner, Lynch, and Selman [54] showed that all these reducibilities are distinct. In Chapter 3 we define reductions between distributional problems.

Given a complexity class $\mathcal{C}$, each reducibility defines a completeness notion in $\mathcal{C}$. Given a reducibility $\le_r^p$, a language L is $\le_r^p$*-complete for* $\mathcal{C}$ if L belongs to $\mathcal{C}$ and every language in $\mathcal{C}$ is $\le_r^p$ reducible to L.

2.4 Almost-Everywhere Hardness

A language L is DTIME$(T(n))$*-complex* (or *does not belong to* DTIME$(T(n))$ *almost everywhere*) [70] if every Turing machine M that decides L runs in time greater than $T(|x|)$, for all but finitely many words x. Geske, Huynh, and Seiferas [27] strengthened the Hartmanis and Stearns hierarchy theorem as follows.

Theorem 2.2. ([27]) *Let T be a fully time-constructible function. Then there is a language L in* $\mathrm{DTIME}(T(n))$ *such that for every function $T'(n)$, if $T'(n)\log(T'(n)) = o(T(n))$, then L is* $\mathrm{DTIME}(T'(n))$*-complex.*

A language L is *immune* to a complexity class $\mathcal{C}$, or $\mathcal{C}$*-immune*, if L is infinite and no infinite subset of L belongs to $\mathcal{C}$. A language L is *bi-immune* to a complexity class $\mathcal{C}$, or $\mathcal{C}$*-bi-immune*, if both L and $\overline{L}$ are $\mathcal{C}$-immune. Balcázar and Schöning [12] observed that for every time-constructible function T, L is $\mathrm{DTIME}(T(n))$-complex if and only if L is bi-immune to $\mathrm{DTIME}(T(n))$.

Given a time bound $T(n)$, a language L is $T(n)$*-printable* if there exists a $T(n)$ time-bounded Turing machine that, on input 0^n, prints all elements of $L \cap \Sigma^n$ [37]. A language L is P*-printable* if there exists a polynomial $p(n)$ such that L is $p(n)$-printable.

A polynomial-time-computable function f is *honest* if there is a polynomial q such that for every y in range(f), there exists x in domain(f) such that $f(x) = y$ and $|x| \leq q(|y|)$. A *one-way* function is an honest polynomial-time computable partial function that is not invertible in polynomial time. Such functions exist if and only if $\mathrm{P} \neq \mathrm{NP}$. Define an honest polynomial-time computable partial function f to be *almost-always one-way* if no polynomial-time Turing machine inverts f correctly on more than a finite subset of range(f).

2.5 Resource-Bounded Measure and Category

Lutz [59, 60] developed resource-bounded measure theory, analogous to classical Lebesgue measure, to study the quantitative structure of complexity classes. Here we give a brief introduction; the interested reader is referred to the published literature [6, 47, 59, 60, 61, 64] for an in depth study. Measures are defined in terms of capital preserving betting strategies called *martingales* [72].

Definition 2.1. A *martingale* is a function $d : \Sigma^* \to [0, \infty)$ with the property that, for all $w \in \Sigma^*$,

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

A martingale d *succeeds* on a language $A \subseteq \Sigma^*$ if

$$\limsup_{n\to\infty} \; d(A(z_0)A(z_1)\cdots A(z_{n-1}))) = \infty,$$

Intuitively, a martingale d is a betting strategy that starts with capital $d(\lambda)$, and bets on the membership or nonmembership of the successive strings $z_0, z_1, \cdots$ in A. Prior to betting on string z_n, the martingale has capital $d(w)$, where $w = A|z_n$. After betting on the string the martingale has capital $d(wb)$ where $b = A(z_n)$. Note that the size of the input to the martingale is exponential in the size of z_n.

Let the classes $p = p_1$ and p_2, both consisting of functions $f : \Sigma^* \to \Sigma^*$, be the classes

$$\begin{aligned} p_1 &= \{f \mid f \text{ is computable in polynomial time}\} \\ p_2 &= \{f \mid f \text{ is computable in } n^{\log n^{O(1)}} \text{ time}\}. \end{aligned}$$

Lutz [59, 60] defined measure on a complexity class as follows.

Definition 2.2. ([59, 60]) A class of languages X has *p_i-measure zero*, denoted $\mu_{p_i}(X) = 0$, if there is a martingale d that belongs to p_i, and d succeeds on every language of X. A class of languages X has *p_i-measure one* if $\mu_{p_i}(\overline{X}) = 0$.

A set X has *measure zero in* E (in EXP) if the p-measure (p_2-measure) of $X \cap$ E ($X \cap$ EXP respectively) is zero. Lutz [59, 60] showed that the measure of P in E is zero and the measure of E in E is one. Lutz hypothesized that "the p-measure of NP is not zero." We call this the "measure hypothesis". This hypothesis seems to have the explanatory power to resolve many questions that have not been resolved by traditional complexity theoretic hypothesis [48, 62, 63]. The measure hypothesis implies the existence of P-bi-immune sets in NP, and the existence of languages that are Turing complete for NP but not many-one complete. These consequences are not known to follow from hypotheses such as "P $\neq$ NP" or "the polynomial-time hierarchy is infinite." Mayordomo [63] and Lutz and Mayordomo [62] showed the following theorems.

Theorem 2.3. ([63]) *The class of* DTIME(2^n)*-bi-immune sets has p-measure one, and therefore, if the p-measure of* NP *is not zero, then* NP *contains a* DTIME(2^n)*-bi-immune set.*

Theorem 2.4. ([62]) *If the p-measure of* NP *is not zero, then* NP *contains a 3-tt-complete language that is not many-one complete.*

Next we define resource-bounded category theory introduced by Ambos-Spies *et al.* [3, 5]. We refer the interested reader to the published literature for details [3, 5, 6, 7]. Here we follow the exposition of Ambos-Spies, Neis, and Terwijn [7].

A *condition* is a set $C \subseteq \Sigma^*$. A *meets* C if for some x, the characteristic sequence $A|x \in C$. C is *dense along* A if for infinitely many strings x there exists $i \in \{0,1\}$ such that the concatenation $(A|x)i \in C$. Then the set A is DTIME($t(n)$)-*generic* if A meets every condition $C \in$ DTIME($t(n)$) that is dense along A. To simplify the notation, we say that A is $t(n)$-*generic* if it is DTIME($t(n)$)-generic.

The "genericity hypothesis" asserts that NP contains a p-generic language. Lutz's measure hypothesis implies the genericity hypothesis, but the converse is not known to be true. The genericity hypothesis also implies the the consequences in Theorems 2.3 and 2.4 [6].

2.6 P-Selective Sets

Selman [77] introduced p-selective sets.

Definition 2.3. ([77]) A language L is *p-selective* if there exists a polynomial-time computable function $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that for all x and y, $f(x,y) \in \{x,y\}$ and if either $x \in L$ or $y \in L$, then $f(x,y) \in L$. We call f a *selector* for L.

Given a finite alphabet Σ, let Σ^ω denote the set of all strings of infinite length of order type ω. For $r \in \Sigma^* \cup \Sigma^\omega$, the standard left cut of r [77, 78] is the set

$$L(r) = \{x \mid x < r\},$$

where $<$ is the ordinary dictionary ordering of strings with 0 less than 1. It is obvious that every standard left cut is p-selective with selector $f(x,y) = \min(x,y)$.

Let L be a p-selective set with selector f. Given any finite set Q, we define a "total preorder" $\leq_f$ on Q ($\leq_f$ is total, reflexive and transitive, but is not necessarily asymmetric) as follows:

$$\begin{aligned} \forall x, y \in Q, x \leq_f y \quad \Leftrightarrow \quad & \exists z_1, z_2, \cdots, z_m \in Q \\ & f(x, z_1) = x, f(z_1, z_2) = z_1, \cdots, \\ & f(z_{m-1}, z_m) = z_{m-1}, f(z_m, y) = z_m. \end{aligned}$$

Note that for any finite set Q we can order the elements of Q in time polynomial of $||Q||$. Let $\perp$ be a special symbol such that $\perp \leq_f x$ for all strings x. Toda [82] proved the following useful property about p-selective sets.

Lemma 2.1. ([82]) *Let L be a p-selective set with selector f. For any finite set Q there exists a string $z \in Q \cup \{\perp\}$ such that $Q \cap L = \{y \in Q \mid y \leq_f z\}$ and $Q \cap \overline{L} = \{y \in Q \mid y >_f z\}$. The string z is called a "pivot" string.*

2.7 Kolmogorov Complexity

We will use the following definitions of Kolmogorov complexity. The book by Li and Vitanyi [57] is an excellent and comprehensive source on Kolmogorov complexity.

Definition 2.4. Fix a universal Turing Machine Φ. For any string x in Σ^*, the *Kolmogorov complexity of x* is defined as

$$K(x) = \min\{|p| \mid \Phi(p) = x\}.$$

We sometimes consider relative Kolmogorov complexity. $K(x|y)$ is the *Kolmogorov complexity of x relative to a string y* where we replace $\Phi(p)$ with $\Phi(p, y)$ in Definition 2.4. $K(x|B)$ is the Kolmogorov complexity of x relative to a set B where we replace $\Phi(p)$ with $\Phi^B(p)$ in Definition 2.4.

Definition 2.5. Fix a universal Turing Machine Φ. Let $t(n)$ be any time bound. For any string x in Σ^*, the *time-bounded Kolmogorov complexity of x* is defined as

$$K^t(x) = \min\{|p| \mid \Phi(p) = x \text{ and } \Phi(p) \text{ outputs } x \text{ in } t(|x|) \text{ steps}\}.$$

Proposition 2.1. *For every n, there is a string x of length n such that $K(x) \geq n$.*

Proposition 2.2. *For every $A \subseteq \Sigma^*$ with $||A|| = m$, there is a string $x \in A$ such that $K(x) \geq \log m$.*

In fact, the following more general statement also holds.

Proposition 2.3. *For any positive integer c and any $A \subseteq \Sigma^*$ with $||A|| = m$, the number of strings $x \in A$ with $K(x) \geq \log m - c$ is at least $m(1 - 2^{-c})$.*

Propositions 2.1, 2.2 and 2.3 hold even when we replace the Kolmogorov complexity $K(x)$ with the time-bounded Kolmogorov complexity $K^t(x)$. They also hold for the relative Kolmogorov complexity $K(x|B)$ for any set B.

Chapter 3

Definition of Average Polynomial Time

3.1 Distributions

Definition 3.1. A *distribution* is a nondecreasing function from Σ^* to the closed interval $[0,1]$ that converges to 1.

Given a distribution μ, the corresponding *density function* μ' is defined by $\mu'(\lambda) = \mu(\lambda)$ and $\mu'(x) = \mu(x) - \mu(x-1)$. For any set of strings S, $\mu(S) = \Sigma_{x \in S} \mu'(x)$.

Given a distribution μ and natural number n, let μ_n denote the conditional distribution of μ on strings of length n. Thus for each $x \in \Sigma^n$,

$$\mu'_n(x) = \begin{cases} \frac{\mu'(x)}{\mu(\Sigma^n)} & \text{if } \mu(\Sigma^n) \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

The conditional distributions $\mu_{\leq n}$ and $\mu_{\geq n}$ are similarly defined on sets $\Sigma^{\leq n}$ and $\Sigma^{\geq n}$. Sometimes we consider distributions over some set S other than Σ^*. In such cases the distribution is a nondecreasing function from S to $[0,1]$.

For a finite set S, the *uniform distribution* assigns $1/||S||$ probability to each element of S. To define uniform distribution on Σ^*, we would want to first randomly pick a length and then select strings of that length. Since it is impossible to select positive integers with equal probability, we cannot hope to define a probability distribution that assigns equal weight to every string of Σ^*. So we define a distribution that is close to uniform distribution.

Definition 3.2. The distribution μ on Σ^* is *uniform*, if for every x, $\mu'(x)$ is equal to $\frac{6}{n^2 2^n \pi^2}$, where $|x| = n$.

Thus, the uniform distribution picks a length n with probability proportional to $1/n^2$. Recall that $\sum 1/n^2 = \pi^2/6$. Note that a distribution that picks a length n with probability $1/2^n$ assigns too much weight to small numbers. This distribution is far from uniform compared to our choice of $1/n^2$. For notational convenience we often omit the normalizing constant $6/\pi^2$, and call $|x|^{-2}2^{-|x|}$ the uniform distribution.

For most of this thesis we restrict our attention to *polynomial-time computable distributions.* Ko [52] defined real-valued functions to be polynomial-time computable as follows.

Definition 3.3. ([52]) A real-valued function $f : \Sigma^* \to [0, 1]$ is *polynomial-time computable* if there is a deterministic polynomial-time-bounded transducer M such that for every string x and every positive integer n,

$$|f(x) - M(x, 1^n)| \le 2^{-n}.$$

Clearly, if μ is computable in polynomial time, then the density function μ' is also computable in polynomial time. Gurevich [32] showed that if $\mathrm{P} \neq \mathrm{NP}$, then the converse is false.

Levin [56] suggested that it is reasonable to restrict attention to distributions that are computable in polynomial time. Many distributions we encounter in practice are polynomial-time computable. For example, the uniform distribution is polynomial-time computable. Levin hypothesized that distributions that arise in nature are either polynomial-time computable or dominated (we define domination in Section 3.3) by a polynomial-time computable distribution.

Ben-David *et al.* [15] considered Levin's suggestion too restrictive. They introduced a broad category of distributions called *p-samplable distributions.*

Definition 3.4. ([15]) A distribution μ is *p-samplable* if there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ that, on input λ, outputs a string x with probability $\mu'(x)$. The running time of $\mathcal{A}$ should be polynomial in $|x|$.

Ben-David *et al.* proved the following theorem.

Theorem 3.1. ([15]) *Every polynomial-time computable distribution is p-samplable. The converse does not hold if one-way functions exist.*

Until now, we considered *infinite distributions*, i.e., distributions on the infinite set Σ^*. Average-case complexity is concerned with the *average* running time of a Turing machine where each input of the machine occurs with some probability. The inputs of the machine come from the sample space Σ^*. Thus, it is justifiable to consider distributions on Σ^*. Alternatively, we can consider a sequence of finite probability distributions on strings of bounded size. For example, we can consider an infinite sequence of distributions where each distribution is over strings of a fixed length.

Definition 3.5. ([46]) Given an infinite sequence of sets $S = (S_1, S_2, \ldots)$, an infinite sequence $\mu = (\mu^1, \mu^2, \cdots)$ is an *ensemble of distributions over* S if each μ^i is a distribution over S_i.

Given a distribution μ over Σ^*, we associate three different ensembles of distributions with μ. The first one is the ensemble $\mu_e = (\mu_1, \mu_2, \cdots)$ where each μ_i is the conditional distribution of μ on Σ^n. The other two ensembles are $\mu_l = (\mu_{\leq 1}, \mu_{\leq 2}, \cdots)$ and $\mu_g = (\mu_{\geq 1}, \mu_{\geq 2}, \cdots)$. Note that μ_g is an ensemble of infinite distributions, whereas μ_e and μ_l are ensembles of finite distributions.

3.2 Levin's Definition

Definition 3.6. *A distributional problem* is a pair (L, μ), where L is a language over Σ^* and μ is a distribution over Σ^*.

Given a distributional problem, we are interested in the question of whether there is a Turing machine that decides L whose running time is polynomial on average, here instances x occur with probability $\mu'(x)$. To make this precise, we should first formulate a notion of what it means for a function $f : \Sigma^* \to \mathbb{N}$ to be polynomial on average.

In worst-case complexity, we define f to be "polynomial" if there exists a constant $k \geq 0$ such that for every n and for every $x \in \Sigma^n$, $f(x) \leq n^k$. A naive way to define average polynomial time is to take the expectation of the function at every length. Thus we are tempted to say that f is (attempt1) polynomial on μ-average if

$$\exists k > 0,\ \forall n\ [\sum_{|x|=n} f(x)\mu'_n(x) \leq n^k]. \tag{3.1}$$

Starting with Levin [56], a number of researchers [15, 32, 88] observed that this definition is neither robust under functional composition of algorithms nor closed under application of polynomials. For example, consider the following function f. Let A^n be a subset of Σ^n with cardinality $2^n - n$.

$$f(x) = \begin{cases} |x|, & \text{if } x \in A^{|x|}, \\ 2^{|x|}, & \text{otherwise.} \end{cases}$$

Consider the distribution $\mu'(x) = \frac{1}{n^2 2^n}$. It is clear that f satisfies Equation 3.1, but f^2 does not. This leads to the following problem. Consider an algorithm that can be simulated in $f(n)$ time on a RAM and in $f^2(n)$ time on a Turing machine. The running time of the algorithm on the RAM is polynomial on average, whereas the running time on the Turing machine is not polynomial on average. Thus the proposed definition (i.e., attempt1) is not machine independent. We cannot build a general theory of average-case complexity using this definition. As an alternative we might propose that a function f is (attempt2) polynomial on μ-average if there exists a positive integer k such that,

$$\forall n \sum_{|x|=n} (f(x))^{1/k} \mu'_n(x) \leq n. \tag{3.2}$$

This proposed definition is closed under the application of polynomials. However, as pointed by Gurevich [32] this definition may be too restrictive. Consider, for example, a function f such that $f(x) = 2^{|x|}$ if $|x|$ is a power of two, and $f(x) = |x|$ otherwise. Consider the distribution μ such that $\mu'(x) = \frac{1}{4^{|x|}}$. Since the value of f is exponential at very few lengths, we would expect f to be polynomial on average with respect to this distribution, but condition (3.2) is not satisfied. This line of reasoning leads us to Levin's definition.

Definition 3.7. ([56]) Given a distribution μ and a function $f : \Sigma^* \to \mathbb{N}$, f is *polynomial on μ-average* if there exists a positive integer k such that

$$\sum_{|x|\geq 1} \frac{(f(x))^{1/k}}{|x|} \mu'(x) < \infty. \tag{3.3}$$

Levin's definition is just as intuitively justified as that is given by equation (3.1). This is seen as follows: The function f is polynomial in worst-case if there exists a constant $k \geq 0$ such that

$$\forall n,\ \forall x \in \Sigma^n\ \ [f(x) < n^k]. \tag{3.4}$$

This is equivalent to

$$\forall n, \forall x \in \Sigma^n\ [\frac{(f(x))^{1/k}}{n} < 1]. \tag{3.5}$$

Levin's definition is obtained by taking the expectation of the above inequality over *all strings* in Σ^*.

Gurevich [32] showed that Levin's definition is closed under composition of algorithms. Thus, Levin's definition is machine independent. Gurevich [32] also showed that Levin's definition and the definition obtained by using equation (3.2) are equivalent for a restricted class of distributions.

Lemma 3.1. ([32]) *If for every n, $\mu(\Sigma^n) = 1/n^l$ for some constant $l > 0$, then a function f is polynomial on μ-average if and only if*

$$\exists k,\ \forall n \sum_{|x|=n} f(x)\mu'_n(x) \leq n^k.$$

Impagliazzo [46] and Wang [88] provide more insight into Levin's definition. Levin's definition is trying to capture the trade-off between a value r and the weight of the instances on which the the value of the function is more than r. The value of the function can be more than super-polynomial only on a sub-polynomial fraction of instances.

Levin's notion of average polynomial time is defined using infinite distributions. Can we define average polynomial time using an ensemble of distributions? Let $\mu = (\mu^1, \mu^2, \cdots)$ be an ensemble of distributions where each μ^i is a distribution over S^i. Consider the following definition:

Definition 3.8. ([46]) A function f is *polynomial on average with respect to the ensemble* μ, if there exists a positive constant k such that for every n, the expectation of $\sum_{x \in S^n}(f(x))^{1/k}/n$ with respect to μ^n is finite.

Is this definition equivalent to Levin's definition? Of course, we cannot directly compare the above definition with Levin's definition, because Levin's definition requires an infinite distribution. However as we noted before, given a distribution μ we can associate with it the ensembles μ_e, μ_l, and μ_g.

Observe that requiring f to be polynomial on average with respect to the ensemble μ_e is the same as requiring equation (3.2) to be true. Thus polynomial on μ-average is not equivalent to polynomial on average with respect to the ensemble μ_e. Impagliazzo [46] observed that we obtain the equivalence if consider the ensemble μ_l.

Lemma 3.2. ([46]) *A function f is polynomial on μ-average if and only if it is polynomial on average with respect to the ensemble $\mu_l = (\mu_{\leq 1}, \mu_{\leq 2}, \cdots)$.*

What happens if we consider the ensemble μ_g? We answer this question in Section 3.5.

Definition 3.9. A distributional problem (L, μ) is solvable in *average polynomial time* if there exists a Turing machine that decides L whose running time is polynomial on μ-average.

Definition 3.10. Average-P is the class of distributional problems that are solvable in average polynomial time.

3.3 Reductions between Distributional Problems

Levin [56] also defined a notion of reductions between distributional problems that enables us to study the relative complexity of distributional problems. Since we want to map distributional problems to distributional problems, mappings must transform distributions as well. We should ensure that frequently occurring instances of one problem are not mapped to rarely occurring instances of the other problem. Otherwise, reductions might map problems that are not in Average-P to problems in Average-P.

Definition 3.11. Let f be a function and μ a distribution on the domain of f. The *induced distribution of f* on $range(f)$, denoted $f(\mu)$, is defined by $f(\mu)'(y) = \mu(\{x \mid f(x) = y\})$.

It is easy to see that $f(\mu)$ is indeed a distribution. Let f be a polynomial-time many-one reduction from A to B. In order for f to reduce (A, μ_A) to (B, μ_B), the induced distribution $f(\mu_A)$ of f should be comparable to the distribution μ_B. The following definitions formalize this idea.

Definition 3.12. ([56, 32]) Let μ_A and μ_B be distributions on Σ^*. The distribution μ_B *dominates* the distribution μ_A, $\mu_A \preceq \mu_B$, if there exists a polynomial p such that for all x, $\mu'_A(x) \leq p(|x|)\mu'_B(x)$.

Definition 3.13. ([56, 32]) Let μ_A and μ_B be distributions on Σ^* and f be a function from Σ^* to Σ^*. Then μ_B *dominates* μ_A *with respect to* f, $\mu_A \preceq_f \mu_B$, if there exists a distribution ν such that $\mu_A \preceq \nu$ and for every $y \in range(f)$, $\mu'_B(y) = f(\nu)'(y)$.

The following definition of reductions is from Levin [56] and Gurevich [32].

Definition 3.14. ([56, 32]) Let (A, μ_A) and (B, μ_B) be two distributional problems. Then (A, μ_A) is *polynomial-time many-one reducible to* (B, μ_B), $(A, \mu_A) \leq^p_m (B, \mu_B)$, if there exists a polynomial-time-computable function $f : \Sigma^* \to \Sigma^*$ such that A is polynomial-time many-one reducible to B via f and $\mu_A \preceq_f \mu_B$. We call the latter condition the *domination condition.*

Gurevich [32] and Wang [88] provide proofs of the following properties.

Theorem 3.2. ([32, 88]) *(i) If* (A, μ_A) *is polynomial-time many-one reducible to* (B, μ_B) *and* $(B, \mu_B) \in$ Average-P*, then* $(A, \mu_A) \in$ Average-P. *(ii) Polynomial-time many-one reductions are transitive.*

It is possible to require only that the reduction be computable in polynomial time on the average [56, 32]: The distribution μ_A is *weakly dominated* by μ_B if there is a function g that is polynomial on the μ_A-average such that for all x, $\mu'_A(x) \leq g(x)\mu'_B(x)$. (A, μ_A) is *many–one reducible to* (B, μ_B) *in average polynomial time*, $(A, \mu_A) \leq^{ap}_m (B, \mu_B)$, if there is a function f that is computable in time a polynomial on the μ_A-average such that A is many–one reducible to B via f and μ_A is weakly dominated by some distribution ν such that for all $y \in range(f)$, $\mu'_B(y) = f(\nu)'(y)$. The analogue of Theorem 3.2 holds for $\leq^{ap}_m$-reductions.

Theorem 3.3. ([32, 88]) *(i) If* (A, μ_A) *is average polynomial-time many-one reducible to* (B, μ_B) *and* $(B, \mu_B) \in$ Average-P*, then* $(A, \mu_A) \in$ Average-P. *(ii) Average polynomial-time many-one reductions are transitive.*

Ben-David *et al.* defined *polynomial-time Turing reductions* between distributional problems.

Definition 3.15. ([15]) Let (A, μ_A) and (B, μ_B) be two distributional problems. Then (A, μ_A) is *polynomial-time Turing reducible to* (B, μ_B), $(A, \mu_A) \leq^p_T (B, \mu_B)$, if there exists a oracle Turing machine M such that $A = L(M^B)$ and there exists a constant $c > 0$ such that for every $y \in \Sigma^*$,

$$\mu'_B(y) \geq \frac{1}{|y|^c} \sum_{x \in Q(y)} \mu'_A(x),$$

where $Q(y)$ is the set of all strings on which the machine M queries y. The latter condition is the *domination condition* for Turing reductions.

We can define *polynomial-time truth-table reductions* between distributional problems similarly. Also, randomized reductions between distributional problems have been defined [15, 32, 85]. In this thesis we restrict our attention to deterministic reductions.

3.4 Distributional NP-Completeness

The class Average-P provides an *easiness* notion that is analogous to the class P. In the same paper, Levin [56] also provided a *hardness* notion, which is analogous to the class of NP-complete problems. Let DistNP be the class of all distributional problems (L, μ) where $L \in \mathrm{NP}$ and μ is a polynomial-time computable distribution.

Definition 3.16. ([56, 32]) A distributional problem (L, μ) *is complete for* DistNP if $(L, \mu) \in \mathrm{DistNP}$ and every distributional problem in DistNP is polynomial-time many-one reducible to (L, μ).

Levin [56] showed that DistNP has complete problems. He proved that a distributional tiling problem is complete for DistNP. Since then several additional DistNP-complete problems have been found [16, 31, 32, 49, 85, 86, 87]. Gurevich [32] showed that the following distributional problem is DistNP-complete.

Let $M_1, M_2, \cdots$ be a fixed enumeration of nondeterministic Turing machines. A triple $\langle i, x, 1^n \rangle$ belongs to the language K if and only if M_i accepts x within n steps. The distribution μ_K is defined as follows:

$$\mu'_K(i, x, 1^n) = \frac{1}{2^{|i|+|x|}} \cdot \frac{1}{|x|^2} \cdot \frac{1}{|i|^2} \cdot \frac{1}{n^2}.$$

Theorem 3.4. ([32, 15]) (K, μ_K) *is complete for* DistNP.

By Theorem 3.2, if a DistNP-complete problem is in Average-P, then $\mathrm{DistNP} \subseteq \text{Average-P}$; i.e., every language in NP is solvable in average polynomial time with respect to every polynomial-time computable distribution. For every NP-complete language L, does there exist a polynomial-time computable distribution μ such that (L, μ) is DistNP-complete? Wang and Belanger [89] showed this is true for all known natural NP-complete problems.

Theorem 3.5. ([89]) *If L is complete for* NP *under honest, one-one, and polynomial-time invertible reductions, then there exists a polynomial-time computable distribution μ such that (L, μ) is* DistNP*-complete.*

Gurevich [32] proved that if a distributional problem (L, μ) is complete for DistNP, then the distribution cannot assign small weight to all instances. A distribution μ is *flat* if $(\exists k > 0)(\forall x)\mu'(x) \le 2^{-|x|^k}$. Some commonly used distributions on graphs are flat and the uniform distribution is flat.

Theorem 3.6. ([32]) *No distributional problem with a flat distribution is complete for* DistNP *unless* NEXP = EXP*.*

"DistNP $\subseteq$ Average-P?" is the average-case analogue of "P = NP?." Ben-David *et al.*[15] gave evidence that the assertion DistNP $\subseteq$ Average-P does not hold.

Theorem 3.7. ([15]) *If* DistNP $\subseteq$ Average-P*, then* E = NE*.*

If P = NP, then DistNP $\subseteq$ Average-P follows trivially. We do not know whether the converse holds. We do not know of any oracle relative to which the converse is false. However, we can show a partial result in this direction. Schuler [73] showed that there exists a language L in EXP such that for every polynomial-time computable distribution μ, $(L, \mu) \in$ Average-P. Schuler's proof relativizes. So for any oracle relative to which NP = EXP (thus P $\neq$ NP) there exists a language L in NP that belongs to Average-P with respect to every polynomial-time computable distribution. If we could arrange for such an L also to be complete for NP under to honest, one-one, and polynomial-time invertible reductions, then, by theorem 3.5, DistNP $\subseteq$ Average-P follows.

Let DistEXP be the class of all distributional problems (L, μ) where $L \in$ EXP and μ is a polynomial-time computable distribution. We can define DistEXP-completeness similar to DistNP-completeness. Let $M_1, M_2, \cdots$ be a fixed enumeration of deterministic Turing machines. A triple $\langle i, x, 1^n \rangle$ belongs to K_E if an only if M_i accepts x within 2^n steps. Define μ_{K_E} to be the same as μ_K. Then (K_E, μ_{K_E}) is complete for DistEXP.

3.5 Cai and Selman's Definition

Levin's definition distinguishes only super-polynomial on average from polynomial on average. Ben-David *et al.* [15] proposed a straightforward generalization to define T on average for an arbitrary time bound T.

Definition 3.17. ([15]) Let μ be a distribution on Σ^*, and let $T : \mathbb{N} \to \mathbb{N}$ be a monotone increasing function. The function $t : \Sigma^* \to \mathbb{N}$ is T *on μ-average* if

$$\sum_x \frac{T^{-1}(t(x))}{|x|}\mu'(x) < \infty.$$

Cai and Selman [19] discovered some limitations of Definition 3.17. Let ATIME($T(n)$) denote the class of distributional problems that are decidable by a Turing machine whose running time is $T(n)$ on μ-average in accordance to this definition. Cai and Selman observed that ATIME(c^n) = ATIME(2^n) for every constant c. This is a weakness of the definition. Cai and Selman argue that a time complexity class should be defined in such a way that it is sufficiently fine to distinguish different inherent time complexities. Another consequence of ATIME(c^n) = ATIME(2^n) is the existence of a DTIME(2^{2n})-complex language that is in ATIME(2^n). Levin's definition has similar problems. When we consider distributions that put most of the weight on few instances, so that the weights on the rest of the instances are negligible, a function larger than every polynomial almost everywhere could still be polynomial on average. For example there exists a P-complex language in DTIME(2^n) that belongs to Average-P with respect to the distribution $\mu'(x) = 1/4^{|x|}$.

To avoid these difficulties, Cai and Selman considered the expectation of the running time of a machine over the set $\Sigma^{\geq n}$, with respect to the conditional distribution $\mu_{\geq n}$. This leads to the following definition:

Definition 3.18. ([19]) Let μ be a distribution on Σ^*, and let

$$W_n = \mu(\{x \mid |x| \geq n\}).$$

Let T be a monotone increasing function from $\mathbb{N}$ to $\mathbb{N}$. Then a function $t : \Sigma^* \to \mathbb{N}$ is *(modified) T on μ-average* if for all $n \geq 1$,

$$\sum_{|x| \geq n} \frac{T^{-1}(t(x))}{|x|}\mu'(x) \leq W_n.$$

Let AVTime($T(n)$) denote the class of all distributional problems (L, μ) that are solvable by a Turing machine whose running time is (modified) T on μ-average. The following result is an easy consequence of the above definition. If a language L is DTIME($t(n)$)-complex then the distributional problem (L, μ) does not belong to AVTime$(t(n))$ for any distribution with infinite support. With this definition, Cai and Selman obtained the following hierarchy theorem.

Theorem 3.8. ([19]) *Let T be fully time-constructible function. Then there is a language L such that for every function $T'(n)$, if $T'(n)\log T'(n) = o(T(n))$, then for every distribution μ the distributional problem (L, μ) belongs to* AVTime$(T(n))$ *but does not belong to* AVTime$(T'(n))$.

It follows from Definition 3.18 that a distributional problem is in average polynomial time if there is Turing machine M that decides L and a positive integer k such that for all $n \geq 1$,

$$\sum_{|x|\geq n} \frac{(T_M(x))^{1/k}}{|x|}\mu'(x) \leq W_n. \tag{3.6}$$

Comparing this with this Levin's requirement (3.3) we see that Cai and Selman's definition requires not only that the infinite sum converges but it converges at a certain rate.

Now the answer to the question we raised in Section 3.2 is obvious. A function is (modified) polynomial on μ average if and only if it is polynomial on μ-average with respect to the ensemble $\mu_g = (\mu_{\geq 1}, \mu_{\geq 2}, \cdots)$.

Definition 3.19. AVP is the class of all distributional problems that are in average polynomial time according Definition 3.18.

It is obvious that Average-P is a subset of AVP. Consider a P-complex (P-bi-immune) language L that belongs to DTIME(2^n). With respect to the distribution $\mu'(x) = 1/2^{2|x|}$, L belongs to Average-P. However, for all distributions μ, (L, μ) does not belong to AVP. Thus AVP is a proper subset of Average-P. Nevertheless, the next theorem shows that the difference between AVP and Average-P is modest.

Definition 3.20. ([19]) A distribution is *reasonable* if there exists a constant $s > 0$ such that $\mu(\{x \mid |x| > n\}) = \Omega(\frac{1}{n^s})$.

Distributions that decrease too quickly (for example $\mu'(x) = 1/2^{2|x|}$) give too much weight to small instances, and therefore are unreasonable. Cai and Selman proved that Levin's definition is equivalent to their definition for reasonable distributions.

Theorem 3.9. ([19]) *Let μ be a polynomial-time computable reasonable distribution. Then $(L, \mu) \in$ AVP if and only if $(L, \mu) \in$ Average-P.*

Next we define distributionally-hard languages.

Definition 3.21. ([19]) A language L is *distributionally-hard* if for every polynomial-time computable distribution μ, the distributional problem (L, μ) does not belong to AVP.

The following theorem follows readily.

Theorem 3.10. ([19]) *Every* P*-bi-immune (*P*-complex) language is distributionally-hard. Thus* EXP *contains distributionally-hard languages.*

Note that this concept is not interesting if we use Levin's definition, because for every language L in EXP there is a distribution μ such that (L, μ) belongs to Average-P.

Schuler and Yamakami [76] considered a notion that in some sense is the opposite of distributionally-hard languages.

Definition 3.22. ([76]) A language L *belongs to* $\mathrm{P}_{\mathrm{pcomp}}$ if (L, μ) belongs to Average-P with respect to every polynomial-time computable distribution.

Schuler [73, 74] and Schuler and Yamakami [75, 76] studied properties of the class $\mathrm{P}_{\mathrm{pcomp}}$. Schuler [73] showed the following containments about $\mathrm{P}_{\mathrm{pcomp}}$.

Theorem 3.11. ([73]) $\mathrm{P} \subsetneq \mathrm{P}_{\mathrm{pcomp}} \subsetneq \mathrm{E}$.

Chapter 4

Reasonable Distributions

Recall that a distribution μ is reasonable if there exists a constant $s > 0$ such that for every n, $\mu(\Sigma^{\geq n}) = \Omega(\frac{1}{n^s})$. Unreasonable distributions put too much weight on small strings, so that the problem we are dealing with becomes essentially a finitary problem. Levin's definition of average polynomial-time is equivalent to Cai and Selman's definition of average polynomial-time, when applied to reasonable distributions. We have seen that Levin's definition yields counterintuitive results when applied to unreasonable distributions; i.e., there is a P-complex language that can be decided in average polynomial-time with respect to some unreasonable distribution.

Cai and Selman's definition of average polynomial-time rectifies this problem. But Cai and Selman's definition has a different shortcoming. Belanger and Wang [14] showed that AVP is not closed under polynomial-time many-one reductions. However, if (A, μ_A) is polynomial-time many-one reducible to (B, μ_B), both μ_A and μ_B are reasonable distributions, and (B, μ_B) belongs to AVP, then (A, μ_A) belongs to Average-P and so, by the equivalence theorem of Cai and Selman, (A, μ_A) belongs to AVP. Thus, if we restrict attention to reasonable distributions, AVP is closed under polynomial-time many-one reductions.

Thus both Levin's definition and Cai and Selman's definition yield unsatisfactory consequences when we permit unreasonable distributions. If we restrict attention to reasonable distributions, then both definitions are equivalent and we do not have these problems. So one simple solution is to restrict one's attention to reasonable distributions.

In this chapter we prove some results that justify this approach. First we improve the result of Belanger and Wang and provide a simpler proof.

We construct a language A and distributions μ and ν such that ν is reasonable, (A, μ) is polynomial-time many-one reducible to (A, ν), (A, ν) belongs to AVP, and (A, μ) does not belong to AVP. In Section 4.2, we study the p-measure of Average-P and AVP and show that Average-P behaves like a feasible class, in terms of resource-bounded measure, only if we restrict attention to reasonable distributions. Also we show that the definition of $\mathrm{P}_{\mathrm{pcomp}}$ remains unchanged under this restriction. In Section 4.3 we provide evidence that the concept of DistNP-completeness is not changed under this restriction. We show that if NP has 2^n-bi-immune sets, then all DistNP-complete problems have reasonable distributions. All distributions throughout this chapter are computable in polynomial time.

4.1 AVP is not Closed under Many-One Reductions

Belanger and Wang [14] showed the following result.

Theorem 4.1. ([14]) *There exist distributional problems (A, μ_A) and (B, μ_B) such that $(A, \mu_A) \leq_m^p (B, \mu_B)$ and (B, μ_B) belongs to* AVP*, but (A, μ_A) does not belong to* AVP*.*

Thus, AVP is not closed under polynomial-time many-one reductions. Here both the distributions μ_A and μ_B are unreasonable. Note that if μ_A were reasonable, then (A, μ_A) would belong to Average-P (since Average-P is closed under polynomial-time many-one reductions) and so, (A, μ_A) would belong to AVP also. So μ_A cannot be a reasonable distribution, but we can ask whether μ_B can be reasonable.

We strengthen the above theorem by showing that μ_B can be reasonable. Our proof also implies that AVP is not closed under domination. Our proof is simpler and has the added advantage of using the identity function for the reduction.

Theorem 4.2. *There exist a language A and distributions μ and ν such that $(A, \mu) \leq_m^p (A, \nu)$, (A, ν) is in* AVP *and ν is a reasonable distribution, but (A, μ) is not in* AVP*.*

Proof. By the almost-everywhere hierarchy theorem [27] (Theorem 2.2), there exists a DTIME(2^n)-complex language L that belongs to DTIME(2^{2n}). Let

$A = L \cup \{1^n : n \geq 1\}$. Notice that if M is a Turing machine that accepts A, then for all but finitely many x in $\Sigma^* - \{1^n\}$, $T_M(x) > 2^{|x|}$.

Define the distributions μ and ν by

$$\mu'(x) = \frac{1}{4^n},$$

$$\nu'(x) = \begin{cases} 1/(n^2 2^n(2^n-1)) & \text{if } x \neq 1^n, \\ (2^n-1)/(n^2 2^n) & \text{if } x = 1^n, \end{cases}$$

where $n = |x|$. Then both μ and ν are polynomial-time computable. Since $\nu(\{x : |x| = n\}) = 1/n^2$, ν is reasonable.

First we show that (A, ν) is in AVP. Since ν is a reasonable distribution, it is sufficient, by Theorem 3.9, to show that (A, ν) is in Average-P. Let M' be a Turing machine that accepts A in time 2^{2n}, and let M be a machine that on input x will accept if $x = 1^{|x|}$, and will simulate M' otherwise. Then

$$\begin{aligned} \sum_x \frac{\sqrt{T_M(x)}}{|x|}\nu'(x) &= \sum_{n=1}^{\infty}\left(\sum_{|x|=n, x\neq 1^n} \frac{\sqrt{T_M(x)}}{|x|}\nu'(x) + \frac{\sqrt{T_M(1^n)}}{|1^n|}\nu'(1^n)\right) \\ &\leq \sum_{n=1}^{\infty} \frac{\sqrt{2^{2n}}}{n} \cdot \frac{1}{n^2 2^n(2^n-1)} \cdot (2^n-1) + \sum_{n=1}^{\infty} \frac{\sqrt{n}}{n} \cdot \frac{2^n-1}{n^2 2^n} \\ &\leq \sum_{n=1}^{\infty} \frac{1}{n^3} + \sum_{n=1}^{\infty} \frac{1}{n^{5/2}} < \infty. \end{aligned}$$

This proves that (A, ν) is in AVP.

We now show that (A, μ) is not in AVP. Notice, first of all, that $W_n = \mu(\{x : |x| \geq n\}) = \sum_{|x|\geq n} 1/4^{|x|} = \sum_{k=n}^{\infty} 1/2^k$. Let M be a Turing machine that accepts A. Then, by construction of A, for sufficiently large n, $T_M(x) > 2^{|x|}$ if $|x| \geq n$ and $x \neq 1^{|x|}$. So, for any $\epsilon > 0$, if n is sufficiently large,

$$\begin{aligned} \sum_{|x|\geq n} \frac{T_M^{\epsilon}(x)}{|x|} \cdot \mu'(x) &\geq \sum_{k=n}^{\infty}\left(\sum_{|x|=k, x\neq 1^k} \frac{T_M^{\epsilon}(x)}{|x|} \cdot \mu'(x)\right) \\ &\geq \sum_{k=n}^{\infty} \frac{2^{\epsilon k}}{k} \cdot \frac{1}{4^k} \cdot (2^k-1) \\ &> \sum_{k=n}^{\infty} \frac{1}{2^k} = W_n. \end{aligned}$$

Hence, (A, μ) is not in AVP.

To complete the proof, we show that $(A, \mu) \leq_m^p (A, \nu)$ via the identity function. It suffices to show that $\mu \preceq \nu$. In particular, we show that for all x, $\mu'(x) \leq |x|^2 \nu'(x)$. To see this, we note that if $x \neq 1^{|x|}$, we have

$$\mu'(x) = \frac{1}{4^{|x|}} \leq \frac{1}{4^{|x|} - 2^{|x|}} = |x|^2 \nu'(x),$$

and if $x = 1^{|x|}$, we have

$$\mu'(x) = \frac{1}{4^{|x|}} \leq \frac{2^{|x|} - 1}{2^{|x|}} = |x|^2 \nu'(x).$$

This completes the proof. □

Since, in the above proof, the many-one reduction is the identity function, it follows that AVP is not even closed under domination.

Theorem 4.3. *The class* AVP *is not closed under domination. There exist a language A and distributions μ and ν such that ν dominates μ and (A, ν) belongs to* AVP*, but (A, μ) does not belong to* AVP*.*

4.2 P-Measure of AP and AVP

We first make a series of simple observations that pinpoint some of the differences between AVP and Average-P. The classes Average-P and AVP are analogous to P. Since the p-measure of P is zero, we would expect Average-P and AVP to have p-measure zero. Similarly, since E − P has p-measure one, we would expect the class of languages that are not in AVP or Average-P to have p-measure one. We do not obtain these expected consequences if we allow unreasonable distributions.

1. Consider the classes

   $$S_1 = \{L \mid \forall \mu, (L, \mu) \notin \text{AVP}\}$$

   and

   $$S_2 = \{L \mid \forall \mu, (L, \mu) \notin \text{Average-P}\}.$$

   S_1 has p-measure one, because every P-complex set belongs to S_1 [19] and the p-measure of P-complex sets is one [63]. For every language L in EXP there exists an unreasonable distribution μ such that $(L, \mu) \in$ Average-P. Thus $S_5 \cap \text{E} = \emptyset$. So the p-measure of S_2 is zero.

2. Consider the classes
$$S_3 = \{L \mid \exists \mu, (L, \mu) \in \mathrm{AVP}\}$$
and
$$S_4 = \{L \mid \exists \mu, (L, \mu) \in \text{Average-P}\}.$$
S_3 has p-measure zero, because $S_3 = \overline{S_1}$, and S_4 has p-measure one, because $E \subseteq S_4$.

Thus Average-P does not yield satisfactory notions of "easiness" and "hardness," in terms of resource-bounded measure. However, we obtain the desired consequences if we restrict our attention to reasonable distributions.

1. Consider the classes
$$S_5 = \{L \mid \forall \text{ reasonable } \mu, (L, \mu) \notin \mathrm{AVP}\}$$
and
$$S_6 = \{L \mid \forall \text{ reasonable } \mu, (L, \mu) \notin \text{Average-P}\}.$$
S_5 has p-measure one, because $S_1 \subseteq S_5$. Since $S_5 = S_6$, S_6 also has p-measure one.

2. Consider the classes
$$S_7 = \{L \mid \exists \text{ reasonable } \mu, (L, \mu) \in \mathrm{AVP}\}$$
and
$$S_8 = \{L \mid \exists \text{ reasonable } \mu, (L, \mu) \in \text{Average-P}\}.$$
The p-measure of S_7 is zero, because $S_7 \subseteq S_3$. Since $S_7 = S_8$, the p-measure of S_8 is also zero.

Thus, when we restrict attention to reasonable distributions we have a meaningful notion of "easiness" and "hardness" in terms of resource-bounded measure.

Next we show that the definition of the class $\mathrm{P}_{\mathrm{pcomp}}$ remains the same under this restriction.

Lemma 4.1. *Let* $(L, \mu) \in \mathrm{AVP}$ *for all reasonable distributions* μ. *Then* $(L, \mu) \in \mathrm{AVP}$ *for all distributions.*

Proof. Let (L, μ) be in AVP for all reasonable polynomial-time computable distributions μ. Let ν be any polynomial-time computable distribution. Now, define a distribution μ as follows:

$$\mu'(x) = \begin{cases} \frac{\nu'(x)}{\nu(\Sigma^n)} \frac{1}{|x|^2} & \text{if } \nu(\Sigma^n) \neq 0, \\ \\ \frac{1}{|x|^2 2^{|x|}} & \text{otherwise.} \end{cases}$$

Clearly, μ is reasonable and polynomial-time computable. Hence, by our assumption, (L, μ) belongs to Average-P. So there exists a positive integer k such that, for all but finitely many n,

$$\sum_x \frac{(T_M(x))^{1/k}}{|x|} \mu'(x) \leq \infty.$$

Since $\mu(\Sigma^n) = \frac{1}{n^2}$, by Lemma 3.1, for every n

$$\sum_{x=n} \frac{(T_M(x))^{1/k}}{|x|} \mu'(x) \leq \frac{1}{n^2}.$$

Multiply both the sides by $\nu(\Sigma^n)n^2$. Since $\mu'(x)\nu(\Sigma^n)n^2 = \nu'(x)$ we obtain the following inequality: For every n,

$$\sum_{|x|=n} \frac{(T_M(x))^{1/k}}{|x|} \nu'(x) \leq \nu(\Sigma^n).$$

Hence for every $n \geq 1$,

$$\sum_{|x| \geq n} \frac{(T_M(x))^{1/k}}{|x|} \nu'(x) \leq \sum_{m \geq n} \nu(\Sigma^n) = \nu(\Sigma^{\geq n}).$$

Thus, $(L, \nu) \in$ AVP. □

If (L, μ) belongs to Average-P for every reasonable distribution, then (L, μ) belongs to AVP for every reasonable distribution, and by previous lemma, (L, μ) belongs to AVP for every distribution. Thus (L, μ) belongs to Average-P for every distribution. We have the following theorem.

Theorem 4.4. *A language L belongs to* $\mathrm{P_{pcomp}}$ *if and only if for every reasonable distribution μ, (L, μ) belongs to* Average-P.

We have the following corollary.

Corollary 4.1. *A language L belongs to* $\mathrm{P}_{\mathrm{pcomp}}$ *if and only if for every distribution μ, (L, μ) belongs to* AVP.

The fact that the class AVP is not closed under many-one reductions raises an interesting question: Does there exist a distributional problem in AVP that is not useful when applied to the concept of completeness? To be specific, "Is there a DistNP-complete problem (L, μ) such that $(L, \mu) \in$ AVP does not imply DistNP $\subseteq$ AVP?" Of course, such distributional problems do not exist if we consider only reasonable distributions. We show that even considering unreasonable distributions such distributional problems do not exist.

Theorem 4.5. *If (L, μ) is* DistNP*-complete and $(L, \mu) \in$* AVP*, then* DistNP $\subseteq$ AVP

Proof. Let (L, μ) be a DistNP-complete problem that is in AVP. Let (L', ν) be any distributional problem in DistNP. Since (L, μ) is DistNP complete, for any reasonable distribution ρ, $(L', \rho) \leq_m^p (L, \mu)$. So, (L', ρ) is in Average-P, hence is in AVP. So, (L', ρ) is in AVP for every reasonable distribution ρ. By Lemma 4.1, L' is in AVP for with respect to every distribution. So $(L', \nu) \in$ AVP. □

4.3 Incompleteness Results

Levin [56] exhibited the first DistNP-complete problem. Since then, several additional DistNP-complete problems have been found [16, 31, 32, 49, 85, 86, 87, 89]. However, we do not possess a catalog of natural DistNP-complete problems that is in any way similar to the flood-tide of NP-complete problems. This distinction is one reason that it is important to analyze individual distributional problems for their potential completeness. It may be useful to understand the structure of the distributions of DistNP-complete problems.

Gurevich showed that if EXP $\neq$ NEXP, then no distributional problem with a flat distribution is complete for DistNP. This was the first incompleteness result. Wang and Belanger [89] showed that if the reductions are restricted to be one-one and honest, then Gurevich's result is true without any assumption. Recall that all known natural NP-complete problems are indeed complete under one-one and honest reductions.

Theorem 4.6. ([89]) *No distributional problems with flat distributions can be complete for* DistNP *under polynomial-time many-one reductions that are one-one and honest.*

Gurevich's result and Wang and Belanger's result hold when we replace polynomial-time many-one reductions with average polynomial-time many-one reductions. Here we show similar incompleteness results. We show that if NP has a DTIME(2^n)-bi-immune language, then all DistNP-complete problems have reasonable distributions. We begin with the following lemma.

Lemma 4.2. *Let μ_1 be the standard uniform distribution, so that $\mu_1(\{x \mid |x| = n\}) = n^{-2}$. Let f be a polynomial-time computable reduction from (A, μ_1) to (B, μ_2), where μ_2 is not reasonable. Then, for all $k \geq 1$, there exist infinitely many strings x, such that $|f(x)|^k \leq |x|$.*

Proof. The function f many-one reduces A to B and $\mu_1 \leq_f \mu_2$. Thus, there exists a distribution ν such that $\mu_1 \preceq \nu$ and for all $y \in range(f)$, $\mu_2'(y) = f(\nu)'(y)$. It is easy to see that ν is reasonable also.

We prove the claim by contradiction. Assume there exist positive integers k and N so that for all strings x, $|x| > N$, $|f(x)|^k > |x|$. We will prove from this assumption that μ_2 is reasonable.

Let $n > N$. Choose s such that $\nu(\{x \mid |x| \geq m\}) = \Omega(m^{-s})$, Consider the following inequalities:

$$\begin{aligned}
\sum_{|z| \geq n^{1/k}} \mu_2'(z) &\geq \sum_{\substack{|z| \geq n^{1/k} \\ z \in f(\Sigma^*)}} \mu_2'(z) \\
&\geq \sum_{\substack{|z| \geq n^{1/k} \\ z \in f(\Sigma^*)}} \sum_{\substack{f(y)=z \\ |y| \geq n}} \nu'(y) \\
&\geq \sum_{|y| \geq n} \nu'(y) \\
&\geq 1/n^s.
\end{aligned}$$

Thus, for all $m \geq N^{1/k}$,

$$\sum_{|z| \geq m} \mu_2'(z) \geq 1/m^{ks},$$

which proves that μ_2 is reasonable. □

Theorem 4.7. *If* $(A, \mu_1) \leq_m^p (B, \mu_2)$*, where* $B \in \mathrm{NP}$*,* μ_1 *is the standard uniform distribution, and* μ_2 *is not reasonable, then* A *is not* $\mathrm{DTIME}(2^n)$*-bi-immune.*

Proof. Let f be a polynomial-time reduction from (A, μ_1) to (B, μ_2), and choose $l \geq 1$ such that $B \in \mathrm{DTIME}(2^{n^l})$.

For all strings x, $x \in A$ if and only if $f(x) \in B$. Membership of $f(x)$ in B can be decided in $2^{|f(x)|^l}$ steps. By Lemma 4.2, for infinitely many strings x, $|f(x)|^l \leq |x|$. Thus, membership in A of these infinitely many strings can be decided in $2^{|x|}$ steps. Hence, A is not $\mathrm{DTIME}(2^n)$-bi-immune. □

Our main result is a corollary of the above theorem.

Theorem 4.8. *If* NP *contains a* $\mathrm{DTIME}(2^n)$*-bi-immune set, then every* $\leq_m^p$*-complete distributional problem for* DistNP *has a reasonable distribution.*

If the p-measure of NP is not 0, then NP has 2^n-bi-immune sets [63]. This gives the following corollary.

Corollary 4.2. *If the* p*-measure of* NP *is not* 0*, then every* $\leq_m^p$*-complete distributional problem for* DistNP *has a reasonable distribution.*

We also obtain these results for $\leq_m^{ap}$-reducibility.

Theorem 4.9. *If* NP *contains a* $\mathrm{DTIME}(2^n)$*-bi-immune set, then every* $\leq_m^{ap}$*-complete distributional problem for* DistNP *has a reasonable distribution.*

Proof. Let (L, μ) be an $\leq_m^{ap}$-complete distributional problem for DistNP. Define the distribution μ_1 by $\mu_1'(0^n) = n^{-2}$, for all $n \geq 1$, and $\mu_1'(x) = 0$, for all $x \notin \{0\}^*$. For all n, $\mu_1(x \mid |x| = n\}) = n^{-2}$, so, by definition, μ_1 is a reasonable distribution. Let $S \in \mathrm{NP}$; choose $l \geq 1$ such that $S \in \mathrm{DTIME}(2^{n^l})$. Let f be a function that is computable in time a polynomial on the μ_1-average and that $\leq_m^{ap}$-reduces (S, μ_1) to (L, μ). By Lemma 3.1, there is a Turing machine M that computes f whose running-time T_M, for some integer $j \geq 1$, satisfies the following inequality, for all $n \geq 1$:

$$\sum_{|x|=n} \frac{T_M(x)^{1/j}}{n} \mu_1'(x) \leq n^{-2}.$$

Thus,

$$\frac{T_M(0^n)^{1/j}}{n} n^{-2} \leq n^{-2},$$

from which it follows that $T_M(0^n) \leq n^j$, for all n. Thus, the restriction of f to $\{0\}^*$ is polynomial-time computable.

Similar to Lemma 4.2, our first task is to demonstrate that for all $s \geq 1$, there exist infinitely many $n \geq 1$ such that $|f(0^n)|^s \leq n$. Let ν weakly dominate μ_1 so that for all strings $y \in range(f)$, $\mu'(y) = f(\nu)'(y)$. There is a function g that is polynomial on the μ_1-average so that for all x, $\mu_1'(x) \leq g(x)\nu'(x)$. As in the previous paragraph, since μ_1 is reasonable, there exists $j \geq 1$ such that for all $n \geq 1$,

$$\sum_{|x|=n} \frac{g(x)^{1/j}}{n} \mu_1'(x) \leq n^{-2},$$

from which, as above, $g(0^n) \leq n^j$. Then,

$$\begin{aligned} \nu(\{x \mid |x| = n\}) &= \sum_{|x|=n} \nu'(x) \\ &\geq \sum_{|x|=n} \mu_1'(x)/g(x) \\ &\geq (n^{-2})(n^{-j}). \end{aligned}$$

It follows readily that ν is reasonable also. Now the proof of our task proceeds exactly as does the proof of Theorem 4.7 and we conclude that S is not DTIME(2^n)-bi-immune.

□

Corollary 4.3. *If the p-measure of* NP *is not* 0*, then every* $\leq_m^{ap}$*-complete distributional problem for* DistNP *has a reasonable distribution.*

Since $\leq_m^p$ is stronger than $\leq_m^{ap}$, Theorem 4.8 and Corollary 4.2 also follow from Theorem 4.9, but Theorem 4.7 is of independent interest.

We obtain unconditional results if we consider honest many-one reductions.

Theorem 4.10. *No distributional problem with an unreasonable distribution is complete for* DistNP *under honest polynomial-time many-one reductions.*

Proof. Let (L, ν) be a DistNP-complete problem with respect to honest polynomial-time many-one reductions, and ν be an unreasonable distribution. Consider the distributional problem (A, μ_1), where μ_1 is the standard uniform distribution and A is a language in NP. Assume f is the honest polynomial-time computable function that reduces (A, μ_1) to (L, ν). Since f is honest, there exists a constant k such that for all x, $|f(x)|^k > |x|$. This contradicts Lemma 4.2. Thus (L, ν) is not complete for DistNP under honest polynomial-time many-one reductions. $\square$

The similar claim holds for $\leq_m^{ap}$ reductions.

Theorem 4.11. *No distributional problem with an unreasonable distribution is complete for* DistNP *under honest average polynomial-time many-one reductions.*

We conclude this chapter by mentioning some open problems concerning reasonable distributions. We have shown that the class $\mathrm{P}_{\mathrm{pcomp}}$ does not change when distributions are restricted to reasonable distributions. It would be interesting to know whether the class of distributionally-hard languages also does not change under this restriction. Gurevich [32] and Venkatesan and Levin [85] defined a notion of randomized reductions between distributional problems. They exhibited some distributional problems with flat distributions that are complete for DistNP under randomized reductions. So it is natural to ask whether every distributional problem that is complete for DistNP, under randomized reductions, should have a reasonable distribution. We have not been able to extend our incompleteness results to randomized reductions. In this regard, we note that all known distributional problems complete for DistNP, under randomized reductions, have reasonable distributions.

Chapter 5

Distributionally-Hard Languages

Consider the fundamental question of what it means for a language L to be difficult to recognize. Usually we identify hard languages with those that do not belong to P. This is *worst-case hardness*. We identify *almost-everywhere hardness* with P-complex languages (equivalently P-bi-immune languages) [70]. Almost-everywhere hardness implies worst-case hardness, but the converse is not true. There are languages that are very easy to recognize on most strings that do not belong to P.

Cai and Selman [19] introduced a notion of *distributional hardness*. Recall that a language L is *distributionally-hard* if for every polynomial-time computable distribution μ, (L, μ) does not belong to AVP. Again, it is easy to see that distributional hardness implies worst-case hardness, but the converse is not true. Cai and Selman showed that every P-bi-immune language (P-complex language) is distributionally-hard. So almost-everywhere hardness implies distributional hardness.

In this chapter we consider the question of whether distributional hardness implies almost-everywhere hardness. We investigate the question of whether there exist distributionally-hard languages that are not P-bi-immune. We show that such languages exist if and only if P contains a P-printable-immune set. Then we extend this characterization significantly to include assertions about several traditional questions about immunity, about finding witnesses for NP-machine, and about existence of one-way functions. Also we study the question of whether NP contains distributionally-hard languages.

5.1 Existence of Distributionally-Hard Languages

Recall that a set A is P-*printable* if there exists a $k \geq 1$ such that a deterministic Turing machine can print all strings of length n that belong to A in time n^k [37, 34]. A set A is P-*printable-immune* if it is infinite and no infinite subset of A is P-printable.

We first say a few words about the assertion "P has a P-printable-immune set." Since every non-sparse set is not P-printable, in order to understand the structure of non-P-printable sets it is natural to ask whether each such set has an infinite P-printable subset. Allender and Rubinstein [2] answered this question by showing that for every non-P-printable set S and time-bound $T(n)$ that majorizes every polynomial, there is a subset A of S in DTIME($T(n)$) that has a finite intersection with every P-printable set. This implies that A is P-printable-immune. It is natural to ask, especially in the case that S belongs to P, whether S has a P-printable-immune subset A that belongs to P. A weaker form of this question is whether P contains P-printable-immune sets.

Theorem 5.1. *There exist distributionally-hard sets that are not* P*-bi-immune if and only if* P *contains a* P*-printable-immune set.*

Proof. Let L be a distributionally-hard set that is not P-bi-immune. Since L is not P-bi-immune, some infinite set S in P either is a subset of L or of $\overline{L}$. Consider the case that $S \subseteq L$. Supposing that S is not P-printable-immune, let S' be an infinite P-printable subset of S. Define μ as follows: For each length n for which S' contains strings of length n, determine the strings $x_1, \ldots, x_{k(n)}$ of length n that belong to S', and define

$$\mu'(x_1) = \cdots = \mu'(x_{k(n)}) = \frac{1}{k(n)} \frac{1}{n^2}.$$

All other strings have weight 0. (It follows that $\mu(S') = 1$ and $\mu(\overline{S'}) = 0$.) Define M to be a Turing machine that first behaves like a P-acceptor for S' and then, on words that the P-acceptor does not accept, behaves like a Turing machine that accepts L. Since $S' \subseteq L$, M accepts L, and it is easy to see that T_M is polynomial on the μ-average, hence L is not distributionally-hard.

To prove the converse, let $B \in$ P be P-printable-bi-immune. Let A be any set that is DTIME(2^{n^3})-complex. We define $L = A \cup B$. Note that A and B

are not disjoint since A is DTIME(2^{n^3})-bi-immune. Since $B \in$ P, clearly, L is not P-bi-immune. Now our goal is to prove that L is distributionally-hard to recognize. Observe that every Turing machine that recognizes L takes more than 2^{n^3} time on all but finitely many strings of $\overline{B}$. Let $u_n = \mu(\{x \mid |x| = n\})$. We require the following lemma.

Lemma 5.1. *Suppose that μ is a distribution such that (L, μ) is in* AVP*. Then, there exist infinitely many n such that $u_n \neq 0$ and*

$$\mu(\{x \mid x \in \overline{B}, |x| = n\}) \leq \frac{nu_n}{2^{n^2}}.$$

Proof. We prove the claim by contradiction. Let $X_n = \{x \mid x \in \overline{B}, |x| = n\}$. Let N be a positive integer such that for all $n > N$, $u_n \neq 0$ and

$$\mu(X_n) > \frac{nu_n}{2^{n^2}}.$$

We will prove that (L, μ) is not in AVP. Let M be any Turing machine that accepts L, let T_M denote the running time of M, and assume that N is sufficiently large so that $T_M(x) > 2^{|x|^3}$ for all strings $x \in \overline{B}$, $|x| \geq N$. Let $k \geq 1$ be any positive integer. Consider the following inequalities.

$$\begin{aligned}
\sum_{|x|>N} \frac{T_M^{1/k}(x)\mu'(x)}{|x|} &\geq \sum_{\substack{|x|>N \\ x\in\overline{B}}} \frac{T_M^{1/k}(x)\mu'(x)}{|x|} \\
&\geq \sum_{\substack{m>N \\ u_m\neq 0}} \sum_{\substack{|x|=m \\ x\in\overline{B}}} \frac{T_M^{1/k}(x)\mu'(x)}{|x|} \\
&> \sum_{\substack{m>N \\ u_m\neq 0}} \sum_{\substack{|x|=m \\ x\in\overline{B}}} \frac{(2^{m^3})^{1/k}\mu'(x)}{m} \\
&\geq \sum_{\substack{m>N \\ u_m\neq 0}} \frac{(2^{m^3})^{1/k}\mu(X_m)}{m} \\
&> \sum_{\substack{m>N \\ u_m\neq 0}} \frac{(2^{m^3})^{1/k}}{m} \frac{mu_m}{2^{m^2}} \\
&> \sum_{\substack{m>N \\ u_m\neq 0}} u_m = \sum_{m>N} u_m
\end{aligned}$$

Thus $(L, \mu) \notin$ AVP. □

Continuing with the proof of Theorem 5.1, next we show that $(L, \mu) \notin$ AVP, for every polynomial-time computable distribution μ. Again, by contradiction, suppose that μ is a polynomial-time computable distribution such that $(L, \mu) \in$ AVP.

Define an *interval* $[x_1, x_2]$ to be a finite sequence of strings in increasing order that begins with the string x_1 and ends with the string x_2. (If we identify every string with the number it represents in dyadic notation, then lexicographic order of strings and the natural ordering of the positive integers coincide.) For example, the set of all strings of length n is the interval $[0^n, 1^n]$. Given strings x_1 and x_2 such that x_1 precedes x_2, let $mid(x_1, x_2) = (x_1 + x_2)/2$. Then, $[x_1, mid(x_1, x_2)]$ contains the first $(x_2 - x_1 + 1)/2$ strings in $[x_1, x_2]$, and $[mid(x_1, x_2) + 1, x_2]$ contains the last $(x_2 - x_1 + 1)/2$ strings in $[x_1, x_2]$. We will use the following programming variables to simplify notation: Given an interval $I = [x_1, x_2]$, "Left_I" denotes the interval $[x_1, mid(x_1, x_2)]$, and "Right_I" denotes the interval $[mid(x_1, x_2) + 1, x_2]$. We define a set T to contain at most one string of length n by the following algorithm:

Current := $[0^n, 1^n]$;
For $i = 1$ **to** n **do**
  if $\mu(\mathrm{Left}_{\mathrm{Current}}) \geq \mu(\mathrm{Right}_{\mathrm{Current}})$
    then Current := $\mathrm{Left}_{\mathrm{Current}}$ **else** Current := $\mathrm{Right}_{\mathrm{Current}}$.

The final value of Current contains exactly one string x. Put x into T if and only if $x \in B$.

Next we will prove that T is an infinite P-printable subset of B, which will complete the proof of Theorem 5.1. Obviously, T is a subset of B. Since μ is computable in polynomial time, $\mu(\mathrm{Left}_{\mathrm{Current}})$ and $\mu(\mathrm{Right}_{\mathrm{Current}})$ can be computed in polynomial time. Thus, T is P-printable.

We need only to show that T is an infinite set. If x is the final value of Current, $|x| = n$, then by the construction, $\mu'(x) \geq u_n/2^n$. However, by Lemma 5.1, there exist infinitely many n such that $u_n \neq 0$ and $\mu(X_n) \leq \frac{u_n n}{2^{n^2}}$. Thus, for all such n, $\mu'(x)$ is greater than $\mu(X_n)$. Hence, for all such n, the final value of Current belongs to B. Thus, T is an infinite set.

This completes the proof. □

It is easy to see that every P-bi-immune set is P-printable-immune. So, if NP has a P-bi-immune set, then NP has a P-printable-immune set. If NP has a P-printable-immune set, then, by a result that Allender and Rubinstein [2] attribute to D. Russo, P has a P-printable-immune set.

Proposition 5.1. *If* NP *has a* P*-bi-immune set, then there is a language* L *that is distributionally hard to recognize but not* P*-bi-immune.*

Note that the language L that we constructed in Theorem 5.1 belongs to EXP. Can we put this language into a smaller time complexity class? Let $t(n)$ grows faster than any polynomial. Can we construct a distributionally-hard language that belongs to DTIME($t(n)$), but not P-bi-immune? Observe that the proof of Theorem 5.1 crucially depends on the exponential bi-immunity of A. However, under a stronger assumption we can obtain the desired result.

Theorem 5.2. *If* UP *contains a* P*-bi-immune language, then (i) For every time constructible function* $t(n)$ *where* $t(n)$ *grows faster than any polynomial, there is a distributionally-hard language in* DTIME($t(n)$) *that is not* P*-bi-immune, and (ii)* UP *has a distributionally-hard language that is not* P*-bi-immune.*

Proof. Let S be a P-bi-immune language in UP. Then $S' = S \cap 0^*$ is a tally P-immune language in UP. Let R be the polynomial-time computable relation associated with the language S'. Define

$$B = \{\langle 0^n, w\rangle \mid R(0^n, w) \text{ holds}\}.$$

It is easy to see that B is a P-printable-immune language in P. Moreover, B has at most one string at each length. Now the proof proceeds similar to the proof of Theorem 5.1. Let, $L = A \cup B$, where A is a P-bi-immune language in DTIME($t(n)$). We obtain the following weaker version of Lemma 5.1, whose proof is similar to the proof of Lemma 5.1.

Lemma 5.2. *Suppose that* μ *is a distribution such that* (L, μ) *is in* AVP. *Then there exist infinitely many* n *such that* $u_n \neq 0$ *and*

$$\mu(\{x \mid x \in \overline{B}, |x| = n\}) \leq \frac{u_n}{2}.$$

So for infinitely many n,

$$\mu(\{x \mid |x| = n, x \in B\}) > \frac{u_n}{2}.$$

B has at most one string at each length. Thus, for infinitely many n, there is a string x such that $\mu'(x) > u_n/2$, and x belongs to B. We can find x by a binary search procedure similar to the one in the proof of Theorem 5.1. Thus, we can exhibit a P-printable subset of B. The proof of item (ii) is similar. $\square$

Let S be a P-printable set. Let $r(x)$ be the number of strings in S that are lexicographically less than x. Consider the following distribution μ that puts its entire weight on S:

$$\mu'(x) = \begin{cases} \frac{1}{(r(x)+1)^2} & \text{if } x \in S, \\ 0 & \text{otherwise.} \end{cases}$$

Since S is P-printable, μ is computable in polynomial time. Now if L has a P-printable subset S, then we can define a distribution μ that puts its entire weight on S. Thus $(L, \mu) \in$ AVP. Similarly if $\overline{L}$ has a P-printable subset, then we can define a polynomial-time computable distribution μ such that $(L, \mu) \in$ AVP. Thus, every distributionally-hard set is P-printable-bi-immune. For any set L, we have the following implications:

$$\begin{aligned} L &\text{ is P-bi-immune} \\ &\Rightarrow \qquad (5.1) \\ L &\text{ is distributionally-hard} \\ &\Rightarrow \qquad (5.2) \\ L &\text{ is P-printable-bi-immune.} \end{aligned}$$

By Theorem 5.1, if P does not have P-printable-immune sets, then implication 5.1 becomes an equivalence. We can show that if P does not have P-printable-immune sets, then implication 5.2 also becomes an equivalence: For this, we show that if P does not have P-printable-immune sets, then every P-printable-bi-immune set is P-bi-immune. Assume S is a P-printable-bi-immune set that is not P-bi-immune. So either S or $\overline{S}$ has an infinite subset L in P. Consider the case that $L \subseteq S$. Since P does not have P-printable-immune sets and $L \in$ P, L has a P-printable subset. This contradicts the P-printable-bi-immunity of S. The case $L \subseteq \overline{S}$ is similar.

Theorem 5.3. *If* P *does not have* P*-printable-immune sets, then a language L is distributionally-hard if and only if L is* P*-printable-bi-immune.*

5.2 Extending the Characterization

In this section, we significantly extend Theorem 5.1 to include assertions about several previously studies hypotheses. The results in this section are obtained in a joint work with Fortnow and Selman [25].

Theorem 5.4. *The following assertions are equivalent.*

1. *There exists a distributionally-hard set that is not* P*-bi-immune.*
2. *There exists a* P*-printable-bi-immune set that is not* P*-bi-immune.*
3. *There exists a set that is* P*-printable-bi-immune, not* P*-bi-immune, but whose complement is* P*-immune.*
4. P *contains a* P*-printable-immune set.*
5. NP *contains a* P*-printable-immune set.*
6. *There is an infinite set S in* NE *and an* NE*-machine M that accepts S such that no* E*-machine correctly computes infinitely many accepting computations of M.*
7. *There is an infinite tally language L in* NP *and an* NP*-machine M that accepts L such that no* P*-machine correctly computes infinitely many accepting computations of M.*
8. *There is an infinite set S in* NP *and an* NP*-machine M that accepts S such that no* P*-machine correctly computes infinitely many accepting computations of M.*
9. *Almost-always one-way functions exist.*

Theorem 5.1 gives the equivalence of items 1 and 4. The equivalence of items 4 and 5 is due to Russo [2]. Also, Theorem 5.4 strengthens a result of Hemaspaandra, Rothe, and Wechsung [41], which is that item 8 implies item 4.

Recall that every language in the class E (NE) is identifiable with a tally language in P (NP), respectively [17]. This observation yields the fact that items 6 and 7 are equivalent. Similarly, NP contains a P-immune tally language if and only if NE contains an E-immune set. One does not expect the existence of a P-immune set in NP to imply the existence of a P-immune tally language in NP, because in general such downward separations do not hold. Hemaspaandra and Jha [40] support this contention with an oracle relative to which NP contains a P-immune set but no P-immune tally language. Thus, the equivalence of items 7 and 8 is surprising and demonstrates that search problems and decision problems have different properties. The following completes the proof.

Proof. As we just noted, items 1, 4, and 5 are equivalent, and items 6 and 7 are equivalent. Items 8 and 9 are equivalent by standard techniques [79].

We will prove the following cycles:

(i) $4 \Rightarrow 7 \Rightarrow 8 \Rightarrow 4$.

(ii) $4 \Rightarrow 3 \Rightarrow 2 \Rightarrow 4$.

Since item 4 appears in each cycle, this will complete our proof. The following lemmas accomplish our task.

Lemma 5.3. $4 \Rightarrow 7 \Rightarrow 8 \Rightarrow 4$.

Proof. Let $A \in \mathrm{P}$ be P-printable-immune. Define

$$L = \{1^n \mid \exists x[|x| = n \text{ and } x \in A]\}.$$

Then, L is an infinite tally language in NP. Consider the NP-machine M that on input 1^n, guesses a string x of length n and accepts if $x \in A$. If some P-machine computes infinitely many accepting computations of M, then A is not P-printable immune. Thus, item 4 implies item 7. Item 7 implies item 8 trivially.

Now we show that item 8 implies item 4. Let S be an infinite set in NP and let M be an NP-machine that accepts S such that no P-machine correctly computes infinitely many accepting computations of M, Define

$$C = \{\langle x, y\rangle \mid \ y \text{ is an accepting computation of } M \text{ on } x\}.$$

Then, $C \in \mathrm{P}$. Let $p(n)$ be a polynomial such that for every $\langle x, y\rangle$ in C, $|\langle x, y\rangle| \leq p(|x|)$. If C has an infinite P-printable subset A, then the following procedure computes accepting computations for infinitely many inputs of M.

input x;
print $A^{\leq p(|x|)}$;
if for some y, $\langle x, y\rangle$ is printed **then** output the first such y **else** reject.

Thus, item 8 implies item 4 □

Lemma 5.4. $4 \Rightarrow 3 \Rightarrow 2 \Rightarrow 4$.

Proof. First we show that item 4 implies item 3. Let $L \in$ P be P-printable immune. Let X be any set that is P-bi-immune. Define $A = X \cup L$. Since L is an infinite subset of A that belongs to P, A is not P-bi-immune. Let B be an infinite set in P. Noting that $\overline{A} \subseteq \overline{X}$, B is not a subset of $\overline{A}$, because X is P-bi-immune. Thus, $\overline{A}$ is P-immune.

Now we show that A is P-printable–bi-immune. Note that we need only to prove that A includes no infinite P-printable set. Let B denote an infinite P-printable set and suppose that B is a subset of A. B does not have a finite intersection with L because if so, then $B \cap \overline{L}$ is an infinite subset of X that belongs to P. Thus, $B \cap L$ is an infinite set. However, $B \cap L$ is P-printable, which is a contradiction because L is P-printable-immune.

Item 3 implies item 2 trivially. We need only to show that item 2 implies item 4. Let A be P-printable-bi-immune but not P-bi-immune. Let L be an infinite set in P so that either $L \subseteq A$ or $L \subseteq \overline{A}$. If L has an infinite P-printable subset, then so does either A or $\overline{A}$. Thus, L is P-printable-immune. □

Theorem 5.4 follows immediately from Lemmas 5.3 and 5.4. □

Recall that a language L is distributionally-hard if for every polynomial-time computable distribution μ, (L, μ) does not belong to AVP. If we extend the definition to insist that (L, μ) does not belong to AVP for all P-samplable distributions, then we do not know whether the new definition coincides with the existing one. Nevertheless, with the new definition, we can still enquire whether there exist such distributionally-hard sets that are not P-bi-immune. Here we offer some partial results concerning this question.

In order to obtain analogues of some of the implications of Theorem 5.4 we need a slightly different definition of P-samplable distributions. Define a distribution μ to be P-*samplable* if there exists a probabilistic polynomial-time-bounded Turing machine M that on input 0^n outputs x with probability $\mu'_n(x) = \mu'(x)/\mu(\Sigma^n)$. Define a set S to be R-*hittable* if there exists a probabilistic polynomial time Turing machine M and a polynomial $p(n)$ such that for infinitely many n,

$$Pr[M(0^n) \in S \cap \Sigma^n] \geq 1/p(n).$$

A language L is *samp-distributionally-hard* if for every P-samplable distribution μ, $(L, \mu) \notin$ AVP. An honest, partial function is *probabilistic-almost-always one-way* if f is computable in polynomial time and, for every polynomial p, for every probabilistic polynomial-time machine D, and for all but finitely many $y \in Range(f)$,

$$\Pr[D(y) \in f^{-1}(y)] < \frac{1}{p(|y|)}.$$

Here the probability is taken over the coin tosses of D.

Theorem 5.5. *If* P *contains a set that is not* R*-hittable, then each of the following assertions hold.*

1. *There exists a samp-distributionally-hard set that is not* P*-bi-immune.*
2. *Probabilistic-almost-always one-way functions exist.*
3. *There is an infinite tally language L in* NP *and an* NP*-machine M that accepts L such that for every polynomial $p(n)$, no probabilistic polynomial time machine correctly computes infinitely many accepting computations of M with probability greater than $1/p(n)$*

Proof. The proof of the first implication is similar to the proof of Theorem 5.1. Let, $L = A \cup B$, where is A is a DTIME(2^{n^3})-complex set and B is a set in P that is not R-hittable. If (L, μ) is in AVP for some P-samplable distribution, then, by Lemma 5.1,

$$\begin{array}{rcl}
\mu\{x \mid x \in \overline{B}, |x| = n\} & \leq & \dfrac{n\mu(\Sigma^n)}{2^{n^2}} \\
\Rightarrow & & \\
\mu_n(\overline{B}) & \leq & \dfrac{n}{2^{n^2}} \\
\Rightarrow & & \\
\mu_n(B) & > & 1 - \dfrac{n}{2^{n^2}}.
\end{array}$$

Since μ is P-samplable, there exists a probabilistic polynomial-time machine M such that $M(0^n)$ outputs x with probability $\mu'_n(x)$. Since $\mu_n(B) > 1 - n/2^{n^2}$, the output of M belongs to B with probability at least $1 - n/2^{n^2}$. Thus, B is R-hittable.

For the second implication, let S be a set in P that is not R-hittable. Define f by $f(x) = 0^{|x|}$, if $x \in S$, and $f(x)$ is undefined, if $x \notin S$. Note that

$range(f) \subseteq 0^*$. Suppose f is not probabilistic-almost-always one-way. Then there exists a probabilistic machine D and a polynomial $p(n)$ such that for infinitely many n,

$$\Pr[D(0^n) \in f^{-1}(0^n)] \geq 1/p(n).$$

By definition of f, $f^{-1}(0^n) \in S$ and $|f^{-1}(0^n)| = n$. Thus for infinitely many n,

$$\Pr[D(0^n) \in S \cap \Sigma^n] \geq 1/p(n).$$

This implies S is R-hittable.

For the third implication, let S be a set in P that is not R-hittable. Let

$$L \quad = \quad \{0^n \mid \exists x, |x| = n, \text{ and } x \in S\}.$$

Clearly L is in NP. Consider the following nondeterministic machine N for L: N on input 0^n guesses x ($|x| = n$) and verifies $x \in S$. So every accepting computation of N corresponds to a string in S. Assume there exists a probabilistic polynomial-time machine D and a polynomial $p(n)$ such that for infinitely many n,

$$\Pr[D(0^n) = \text{ an accepting computation of } N \text{ on } 0^n] \quad \geq \quad \frac{1}{p(n)}.$$

Since every accepting computation of N on 0^n corresponds to a string in $S \cap \Sigma^n$,

$$\Pr[D(0^n) \in S \cap \Sigma^n] \geq \quad \tfrac{1}{p(n)}.$$

Thus, S is R-hittable. □

We are unable to obtain converses of the above implications. The difficulty in obtaining the converse of the first implication is that we do not know how to define a P-samplable distribution that puts its entire weight on an R-hittable set. We can obtain a converse of the second implication if we define R-hittable sets and probabilistic one-way functions slightly differently.

Define a set S to be *R1-hittable* if there exists a probabilistic polynomial time machine M and a polynomial $p(n)$ such that for infinitely many n there exists a string $x \in S$ such that $|x| = n$, and

$$\Pr[M(0^n) = x] \geq 1/p(n).$$

Note that if S is R1-hittable, then S is R-hittable. An honest partial function is *almost1-always one-way* if f is polynomial-time computable and for every polynomial $p(n)$, for every probabilistic polynomial-time machine D, for all but finitely many y in Range of f,

$$\forall x, f(x) = y, \Pr[D(y) = x] \leq 1/p(n)$$

Note that if f is almost-always one-way, then f is almost1-always one-way. Now, we have the following implication

Proposition 5.2. *If almost1-always one-way functions exist then* P *contains a set that is not R1-hittable.*

Proof. Let f be the given one-way function. Consider the set $S = \{\langle x, f(x)\rangle\}$. Clearly, S belongs to P. If S is R1-hittable, then there exists a probabilistic machine M and a polynomial $p(n)$ such that for infinitely many n, there exists a tuple $\langle x, f(x)\rangle$, such that $|\langle x, f(x)\rangle| = n$, and

$$\Pr[M(0^n) = \langle x, f(x)\rangle] \geq \frac{1}{p(n)}.$$

We define a machine D to invert f as follows. Let y be the input. Since f is honest, there exists a polynomial q such that for every x with $f(x) = y$ $|\langle x, y\rangle| \leq q(|y|)$. Now, D runs M on 0^i, where i ranges from 1 to $q(|y|)$. If M outputs a tuple $\langle x, y\rangle$, then D verifies $f(x) = y$ and outputs x. The following implications show that D correctly inverts y.

$$\exists^\infty n, \exists\langle x, f(x)\rangle, |\langle x, f(x)\rangle| = n, \Pr[M(0^n) = \langle x, f(x)\rangle] \geq \frac{1}{p(n)}$$

$$\Rightarrow$$

$$\exists^\infty y, \exists n \leq q(|y|), \exists x, f(x) = y, |\langle x, y\rangle| = n, \Pr[M(0^n) = \langle x, f(x)\rangle] \geq \frac{1}{p(n)}$$

$$\Rightarrow$$

$$\exists^\infty y, y \in \mathit{range}(f), \exists x, f(x) = y, \Pr[D(y) = x] \geq \frac{1}{p(q((|y|))}.$$

Thus f is not almost1-always one-way. □

Theorem 5.4 tempts us to raise another side issue. Since this theorem gives a nice structural characterization of almost-always one-way functions, we should ask whether a similar characterization of cryptographic one-way functions can be obtained. Although we do not know the answer to this question, from the proof of Theorem 5.4 we obtain an interesting result about one-way permutations.

A polynomial-time-computable, length preserving function f is a *one-way permutation* if f is a bijection and for every probabilistic polynomial-time machine D, every polynomial $p(n)$, and for all but finitely many n,

$$\Pr_{x \in \Sigma^n}[D(f(x)) = x] < \frac{1}{p(n)}.$$

A set S is P-*rankable* [28] if there exists a polynomial-time computable function $rank$ such that for every x, $rank(x)$ is the number of strings in S that are less than x.

A set S has *negligible density* if for every polynomial p and for all n $||S^{=n}|| < 2^n/p(n)$. The following proposition says that the existence of one-way permutations implies the existence of a set S in P that is immune to P-rankable sets of non-negligible density.

Proposition 5.3. *If one-way permutations exist, then there exists a set S in* P *such that every* P*-rankable subset of S has negligible density.*

Proof. Let f be any one-way permutation. Let $S = \{\langle x, y\rangle \mid y = f(x)\}$. Clearly, S belongs to P. Assume S has a P rankable subset S' with non-negligible density. Observe that if a tuple $\langle x, y\rangle$ belongs to S', then the ranks of $\langle 0^{|y|}, y\rangle$ and $\langle 1^{|y|}, y\rangle$ differ by exactly one, because f is one-to-one and f is length preserving. Also observe that if the ranks of $\langle x-1, y\rangle$ and $\langle x, y\rangle$ differ by one, where $x-1$ is the predecessor of x, then $f(x) = y$. Since the density of S' is non negligible, there exists a polynomial $p(n)$ such that for each n, at least $2^n/p(n)$ tuples of the form $\langle x, y\rangle$, belong to S', where $|x| = n$ and $f(x) = y$. Thus for each n, there exists at least $2^n/p(n)$ strings y, where $|y| = n$, such that the ranks of $\langle 0^n, y\rangle$ and $\langle 1^n, y\rangle$ differ by one. Our procedure D, on input y, does a binary search over the interval $[\langle 0^{|y|}, y\rangle, \langle 1^{|y|}, y\rangle]$, until it finds a string x such that the ranks of $\langle x-1, y\rangle$ and $\langle x, y\rangle$, differ by one. Then the procedure D outputs x. Since the density of S' is at least $2^n/p(n)$, the procedure correctly computes the inverse correctly on $2^n/p(n)$ strings for each n. Thus,

$$\Pr_{x \in \Sigma^n}[D(f(x)) = x] \geq \frac{1}{p(n)}.$$

This contradicts the one-way property of f. $\square$

5.3 Does NP Contain Distributionally-Hard Sets?

In this section we study the question of whether NP contains distributionally-hard sets.

If NP contains P-bi-immune sets, then NP contains distributionally-hard sets. Does NP contain a set that is distributionally-hard but not P-bi-immune? The distributionally-hard language L that we constructed in the proof of Theorem 5.1 belongs to EXP. If we assume a stronger hypothesis "NP has a 2^{n^3}-bi-immune set," then L belongs to NP. For this, we define $L = A \cup B$, where A is the 2^{n^3}-bi-immune set that belongs to NP and B is a P-printable immune set belonging to P. Mayordomo [63] showed that if the p_2-measure of NP is not zero, then NP contains 2^{n^3}-bi-immune sets. However, Ambos-Spies *et al.* [8] showed that the p-measure of NP is zero if and only if the p_2-measure of NP is zero. Thus, if the p-measure of NP is not zero, then NP contains 2^{n^3}-bi-immune sets. We have the following corollary.

Corollary 5.1. *If the p-measure of* NP *is not zero, then* NP *contains a distributionally-hard language that is not* P*-bi-immune.*

Relative to a random oracle the p-measure of NP is not zero [51]. So relative to a random oracle NP has 2^{n^3}-bi-immune languages. This gives the following theorem.

Theorem 5.6. *Relative to a random oracle* NP *contains distributionally-hard languages and languages that are distributionally-hard but not* P*-bi-immune.*

Next, we show that if NP has a distributionally-hard language, then NP has a P-immune set.

Theorem 5.7. *If* NP *contains a distributionally-hard set, then* NP *contains a tally P-immune set.*

Proof. Let $L \in$ NP be distributionally hard. We will show that $L \cap \{0\}^*$ is P-immune.

First we argue that $L \cap \{0\}^*$ is an infinite set. Let us suppose otherwise. Then, $\overline{L}$ contains an infinite subset S of $\{0\}^*$ that belongs to P. For each string x in S, let $r(x)$ be the number of strings in S that are lexicographically less than x. Define a distribution μ on Σ^* as follows: $\mu'(x) = (r(x)+1)^{-2}$, for $x \in S$, and $\mu'(x) = 0$, otherwise. A Turing machine that, on input x, first determines whether $x \in S$, accepts if so, and otherwise simulates an acceptor for L, runs in time a polynomial on the μ-average and accepts L. Thus, (L, μ) belongs to AVP, which contradicts our hypothesis. Thus, $L \cap \{0\}^*$ is an infinite set.

Similar argument shows that $L \cap \{0\}^*$ is P-immune, and, of course, $L \cap \{0\}^* \in$ NP. □

If NP contains distributionally-hard languages, then P $\neq$ NP follows trivially. However, there is a relativized world where the converse is not true.

Theorem 5.8. *There is an oracle relative to which* P $\neq$ NP *and* NP *does not contain distributionally-hard languages.*

Proof. Homer and Mass [44] constructed an oracle relative to which P $\neq$ NP and NP does not have P-immune sets. Relative to this oracle, by Theorem 5.7, NP does not contain distributionally-hard languages. □

We summarize the known implications as follows. For any language L in NP,

$$\begin{array}{r} L \text{ is P-bi-immune} \\ \Rightarrow \\ L \text{ is distributionally-hard} \\ \Rightarrow \\ L \cap 0^* \text{ is P-immune} \\ \Leftrightarrow \\ L \cap 0^* \text{ is P-printable-immune} \\ \Rightarrow \\ \text{The assertions of Theorem 5.4 hold.} \end{array}$$

We conclude this chapter by mentioning a few open problems. The distributionally-hard but not P-bi-immune language that we constructed in

Theorem 5.1 is the union of an exponentially-bi-immune language with a P-printable-immune language. Is this the only way to construct such languages? If NP contains P-bi-immune languages, then NP contains distributionally-hard languages. We do not know whether the converse is true. Theorem 5.7 gives a partial answer to this question.

Chapter 6

Comparison of Reductions between Distributional Problems

In this chapter we compare different notions of reductions among distributional problems. In worst-case complexity theory, Ladner, Lynch, and Selman [54] compared the strength of polynomial-time reducibilities. They showed for the common polynomial-time reducibilities, Turing ($\leq_T^p$), truth-table ($\leq_{tt}^p$), bounded truth-table ($\leq_{btt}^p$), and many-one ($\leq_m^p$), that

$$\leq_m^p \prec \leq_{btt}^p \prec \leq_{tt}^p \prec \leq_T^p,$$

where $\leq_r^p \prec \leq_s^p$ means that $\leq_r^p$ is *properly stronger* than $\leq_s^p$; that is, for every A and B, $A \leq_r^p B$ implies $A \leq_s^p B$, but there exist languages A and B such that $A \leq_s^p B$ but $A \not\leq_r^p B$. We show similar results for reductions among distributional problems. Recall that DistEXP is the class of all distributional problems (L, μ) where L belongs to EXP and μ is a polynomial-time computable distribution. We study whether various completeness notions for DistEXP are different. All the distributions in this chapter are polynomial-time computable.

Simon and Gill [80] proved the following theorem.

Theorem 6.1. ([80]) *For each decidable set $A \notin$ P, there is a decidable set B such that $A \leq_{2-tt}^p B$ but $A \not\leq_m^p B$.*

Using the same technique, we can show that there exist distributional problems (A, μ_A) and (B, μ_B) such that $(A, \mu_A) \leq^p_{2-tt} (B, \mu_B)$ but $(A, \mu_A) \not\leq^p_m (B, \mu_B)$.

Theorem 6.2. *Let A be any language that is not in* P *and μ_A be any polynomial-time computable distribution. There exists a distributional problem (B, μ_B) such that $(A, \mu_A) \leq^p_{2-tt} (B, \mu_B)$ but $(A, \mu_A) \not\leq^p_m (B, \mu_B)$.*

Proof. We will construct B in stages. We will ensure that $A \leq^p_{2-tt} B$ by preserving the following condition:

$$x \in A \Leftrightarrow \text{ exactly one of } x0 \text{ and } x1 \text{ belongs to } B. \tag{6.1}$$

The distribution μ_B is defined as $\mu'_B(x0) = \mu'_B(x1) = \mu_A(x)/2$. Clearly, μ_B is polynomial-time computable. On input x, the truth-table reduction queries $x0$ and $x1$ and accepts x if and only if exactly one of $x0$ and $x1$ belongs to B. For every x, $\mu'_A(x) = 2 \times \mu'_B(x0) = 2 \times \mu'_B(x1)$. Thus the reduction satisfies the domination condition. Thus, $(A, \mu_A) \leq^p_{2-tt} (B, \mu_B)$. We now show how to construct B.

Let $f_1, f_2, \cdots$ be an enumeration of polynomial-time computable functions. At stage 0, B is completely undefined, and $n_0 = 0$. At stage i we assume that B is defined on all strings of length less than n_i. We call this set $B(i)$. The construction at stage i consists of the following steps:

1. Find the smallest x such that either

   (a) $|f_i(x)| < n_i$ and $x \in A \Leftrightarrow f_i(x) \notin B(i)$, or

   (b) $|f_i(x)| \geq n$.

   If (a) holds for the x that is found, then define $n_{i+1} = n_i + 1$ and go to the next stage. In this case f_i clearly does not many-one reduce A to B. If (b) holds for the string that is found, then go to step 2.

2. Let $f_i(x) = zb$, where $b \in \{0, 1\}$. If x belongs to A, then place zb in $\overline{B}$ and $z\bar{b}$ in B. If x does not belong to A, then place both zb and $z\bar{b}$ in B. This ensures that Condition 6.1 is satisfied. Define $n_{i+1} = 1 + |f_i(x)|$.

Note that step 1 always finds a string x, for otherwise A is many-one reducible to the finite set $B(n)$, which implies that $A \in$ P. It is clear from the construction the $A \not\leq^p_m B$, thus $(A, \mu_A) \not\leq^p_m (B, \mu_B)$. Since $(A, \mu_A) \leq^p_{2-tt} (B, \mu_B)$ our proof is complete. ☐

Note that the proof of the above theorem relies on the fact that A is not many-one reducible to B. This leads us to the question of whether we can construct distributional problems (A, μ_A) and (B, μ_B) such that $A \leq_m^p B$ and $(A, \mu_A) \leq_{2-tt}^p (B, \mu_B)$ but $(A, \mu_A) \not\leq_m^p (B, \mu_B)$. We show that this can be achieved. We first define a useful function *tower* that we will be using at many places in this chapter. Define $tower(0) = 0$ and $tower(k) = 2^{tower(k-1)}$ for $k > 0$.

Theorem 6.3. *There exist a language A and reasonable distributions μ and ν such that $(A, \mu) \leq_{2-tt}^p (A, \nu)$ but $(A, \mu) \not\leq_m^p (A, \nu)$.*

Proof. Let L be a 2^n-bi-immune language in DTIME(2^{2n}). Let

$$S = \{0^n \mid \exists k,\ n = tower(k)^2\},$$

and $B = L \cap S$. Define the distributions μ and ν as follows.

$$\mu'(x) = \begin{cases} \frac{1}{2^n n^2} & \text{if } \forall k\ |x| \neq tower(k)^2, \\ \frac{1}{n^2} & \text{if } x \in S, \\ 0 & \text{otherwise,} \end{cases}$$

$$\nu'(x) = \begin{cases} \frac{1}{2^n n^2} & \text{if } \exists k\ |x| \neq tower(k), \\ \frac{1}{n^2} & \text{if } x = 0^n \text{ and } \exists k\ n = tower(k), \\ \frac{1}{n^2} & \text{if } x = 0^{n-1}1 \text{ and } \exists k\ n = tower(k), \\ \frac{1}{2^n n^2} & \text{otherwise.} \end{cases}$$

Note that if $x \in S$, then $\nu'(x) = 1/2^n n^2$, so $\nu'(x)$ is much smaller than $\mu'(x)$. We construct the language A in stages. Initially, at stage $k = 0$, $A = B$. Let $f_1, f_2, \cdots$ be an enumeration of polynomial-time computable functions such that each function appears infinitely often in the list. Note that B is a tally set and if $0^n \in B$, then there exists a constant $k > 0$ such that $n = tower(k)^2$. We construct A such that A differs from B only at strings of length n, where $n \in range(tower)$. We also ensure that, for $0^n \in S$, the following condition holds.

$$0^n \in A \Leftrightarrow \text{ exactly one of } 0^{\sqrt{n}} \text{ and } 0^{\sqrt{n}-1}1 \text{ belongs to } A. \tag{6.2}$$

Stage k: Let $n = tower(k)^2$ and $m = \sqrt{n}$. Note that $0^n \in S$. If $|f_k(0^n)| \neq m$ then place exactly one of $0^{\sqrt{n}}$ and $0^{\sqrt{n-1}}1$ into A if and only if $0^n \in A$. This ensures that Condition 6.2 is satisfied. If $|f_k(0^n)| = m$, then we first ensure that f_k is not a many-one reduction from A to A. Let

$y = f_k(0^n)$. If $0^n \in A$, then place y into $\overline{A}$. Put exactly one of $0^m, 0^{m-1}1$ into A. If $0^n \notin A$, then put y, 0^m and $0^{m-1}1$ into A. It is obvious from the construction that f_k does not many-one reduce A to A. It is also clear that 0^n belongs to A if and only if exactly one of 0^m or $0^{m-1}1$ belongs to A.

It is clear from the construction that if a string x belongs to A, then either $x \in S$ or there exists a k such that $|x| = tower(k)$. Given a string x, if $x \in S$, then we can decide whether x belongs to A by deciding whether x belongs to L, and if $|x| = tower(k)$ for some $k > 0$, then we can decide whether x belongs to A by deciding whether $0^{|x|^2}$ belongs to A. Since L is in DTIME(2^{2n}), A is in DTIME(2^{2n^2}). Since $A \cap S = L \cap S$ and L is 2^n-bi-immune, any machine that correctly decides A takes more than 2^n time on all but finitely many strings of S. It is also clear from the construction that for any polynomial-time computable function f, if $|f(x)| = \sqrt{|x|}$ for all but finitely many strings of S, then f does not many-one reduce A to A.

Lemma 6.1. $(A, \mu) \leq^p_{2-tt} (A, \nu)$.

Proof. Given x, if $x \notin S$ and $|x| \notin range(tower)$ then x does not belong to A. If $|x| \in range(tower)$, then the reduction queries x. If $|x| \in range(tower)$, then $\mu'(x) = 1/2^n n^2$ and $\nu'(x) \geq 1/2^n n^2$, where $|x| = n$. Thus the reduction satisfies the domination condition. If $x \in S$ then, $x = 0^n$. In this case, by Condition 6.2, 0^n belongs A if and only if exactly one of $0^{\sqrt{n}}$ and $0^{\sqrt{n}-1}1$ belongs to A. Since $\mu'(0^n) = 1/n^2$ and $\nu'(0^{\sqrt{n}}) = \nu'(0^{\sqrt{n}-1}1) = 1/n$, the domination condition is satisfied. Thus $(A, \mu) \leq^p_{2-tt} (A, \nu)$. □

Lemma 6.2. $(A, \mu) \not\leq^p_m (A, \nu)$.

Proof. Assume (A, μ) is many-one reducible to (A, ν) via f. Since f is polynomial-time computable, there exists a constant $l > 0$ such that $|f(x)| < |x|^l$. Recall that

$$S = \{0^n \mid \exists k > 0, n = tower(k)^2\}.$$

Consider f on the elements of S. Suppose there exist infinitely many x in S such that $|f(x)| > \sqrt{|x|}$. Consider such x $(= 0^n)$. Assume $n = tower(k)^2$. Since f is polynomial-time computable, $|f(x)| < tower(k+1)$. Thus $tower(k) < |f(x)| < tower(k+1)$. Since $\mu'(x) = 1/n^2$ and $\nu'(f(x)) = 1/2^m m^2$, where $|f(x)| = m$, ν can not dominate μ via f.

Now, suppose for infinitely many x in S that $|f(x)| < \sqrt{|x|}$. The following procedure contradicts the fact that any algorithm that correctly decides A

takes more than 2^n time on all but finitely many strings of S. Assume M decides A correctly in time 2^{2n^2}.

input: x
if $x \notin S$
  then simulate M on x
else ($x \in S$, thus $x = 0^n$ and $n = tower(k)^2$)
  if $|f(x)| \geq \sqrt{|x|}$
    then simulate M on x
  else ($|f(x)| < tower(k)$, let $|f(x)| = m$)
    if $m > tower(k-1)^2$ (so $tower(k-1)^2 < m < tower(k)$,
      thus $f(x) \notin A$)
      x does not belong to A
    else ($m \leq tower(k-1)^2$
      run M on $f(x)$ (time taken $= 2^{2m^2} < 2^{2tower(k-1)^4} < 2^{tower(k)^2}$);
      x belongs to A if and only if M accepts $f(x)$.

Thus the only remaining possibility is that for all but finitely many x in S, $|f(x)| = \sqrt{|x|}$. In this case, we have ensured that f is not a many-one reduction from A to A. Thus (A, μ) is not many-one reducible to (A, ν). □

The theorem follows from Lemma 6.1 and Lemma 6.2. □

For every language L and distribution μ, the distributional problem (L, μ) is trivially many-one reducible to (L, μ). Suppose A is many-one reducible to B, then does (A, μ) many-one reduce to (B, μ) for every distribution μ? We show that this is not always true.

Theorem 6.4. *There exists two languages A and B and a reasonable distribution μ such that $A \leq_m^p B$ but $(A, \mu) \not\leq_m^p (B, \mu)$.*

Proof. We construct A and B such that any many-one reduction from A to B must be length increasing infinitely often. We achieve this by diagonalizing against length decreasing reductions. Initially both A and B are empty. Let $f_1, f_2, \ldots$ be an enumeration of polynomial-time computable functions such that each function appears infinitely often in the list. Before stage n, A is defined only on strings of length less than $tower(n)$. Decisions about membership in B have been made for some strings of length less than or equal to $tower(n-1) + n - 1$, where some strings are already placed into B and some strings are already placed into $\overline{B}$.

Stage n: Let $x = 0^{tower(n)}$ and $y = f_n(x)$. If $|y| > |x|$ then skip this stage. From now assume $|y| \leq |x|$. If y already belongs to B, then place x into $\overline{A}$ and $x0^n$ into $\overline{B}$. If y already does not belong to B, then define x to be in A and $x0^n$ to be in B. If B is not defined at y, then put y into B, x into $\overline{A}$, and $x0^n$ into in $\overline{B}$.

It is clear that A is many-one reducible to B via $f(0^{tower(n)}) = 0^{tower(n)+n}$.

Lemma 6.3. *If A is many-one reducible to B via f, then f should be length increasing on infinitely many strings of the form $0^{tower(n)}$.*

Proof. Assume that for all but finitely many string of the form $0^{tower(n)}$ the function f is not length increasing. The function f occurs infinitely often in the enumeration $f_1, f_2, \cdots$. However, whenever $|f(0^{tower(n)})| \leq tower(n)$ we defined A and B such that f is not a many-one reduction from A to B. $\square$

Now we define the distribution μ as follows: $\mu'(0^{tower(n)}) = \frac{1}{n^2}$ and $\mu'(x) = 0$ for all other strings.

Lemma 6.4. *(A, μ) is not many-one reducible to (B, μ).*

Proof. Let f be a many-one reduction from (A, μ) to (B, μ). By the previous claim f should be length increasing on infinitely many strings of the form $0^{tower(n)}$. Since f is polynomial-time computable, there exists a constant l such that $|f(0^{tower(n)})| \leq tower(n)^l$ for all n. So, for all but finitely many n, $f(0^{tower(n)})$ belongs to the interval $(0^{tower(n)}, 0^{tower(n+1)})$. However, for every string x in the interval $(0^{tower(n)}, 0^{tower(n+1)})$, $\nu'(x) = 0$. Since $\mu'(tower(n)) = 1/n^2$, the domination condition is not satisfied. Thus (A, μ) is not many-one reducible to (A, ν). $\square$

The theorem follows from Lemma 6.3 and Lemma 6.4. $\square$

Assume A is truth-table reducible to B via (g, h), where g is the query generator and h is the truth-table evaluator. Let μ_A be any polynomial-time computable distribution. We can define a polynomial-time computable distribution μ_B such that (A, μ_A) is truth-table reducible (B, μ_B) if the following condition holds. For every x,

$$g(x) \quad = \quad \{\langle x, q_1\rangle, \langle x, q_2\rangle, \cdots \langle x, q_k\rangle\} \text{ and } |q_1| = \cdots = |q_k|. \tag{6.3}$$

Define μ_B by

$$\mu'_B(\langle x, q\rangle) = \begin{cases} \mu'_A(x)/k & \text{if } \langle x, q\rangle \in g(x), \\ 0 & \text{otherwise.} \end{cases}$$

It is clear that μ_B is polynomial-time computable. We claim that (A, μ_A) is truth-table reducible to (B, μ_B) via (g, h). A tuple $\langle x, q \rangle$ is queried only on input x. Since $\mu'_B(\langle x, q \rangle) = \mu'_A(x)/k$ and k is bounded by a polynomial, domination condition is satisfied. Thus (A, μ_A) is truth-table reducible to (B, μ_B).

Watanabe [90] showed, for $k \geq 1$, that $\leq^p_{k-tt}$ completeness for EXP is different from $\leq^p_{(k+1)-tt}$ completeness. He exhibited a language B such that K_E is $(k+1)$-truth-table reducible to B, but K_E is not k-truth-table reducible to B. Here K_E is the canonical complete set for EXP. The $(k+1)$-truth-table reduction from K_E to B satisfies Condition 6.3. Recall that (K_E, μ_{K_E}) is complete for DistEXP. Thus, by the previous observation, we can define a distribution μ_B such that (B, μ_B) is $(k+1)$-truth-table complete for DistEXP but not k-truth-table complete. Watanabe also constructed a language A such that K_E is truth-table reducible to A but not bounded truth-table reducible. The truth-table reduction from K_E to A satisfies Condition 6.3. Thus there exists a distributional problem that is truth-table complete for DistEXP but not bounded truth-table complete.

Theorem 6.5. *For $k \geq 1$, $(k+1)$-truth-table completeness is different from k-truth-table completeness for* DistEXP. *Truth-table completeness is different from bounded truth-table completeness for* DistEXP.

We cannot use a similar idea to separate Turing reductions from truth-table reductions for distributional problems. Assume A is Turing reducible to B and μ_A is a polynomial-time computable distribution. Since the queries made by the oracle Turing machine are not known in advance, we do not know how to define a polynomial-time computable distribution μ_B such that μ_B dominates μ_A via the Turing reduction. Nevertheless, we can separate Turing reductions from truth-table reductions by a different idea.

Theorem 6.6. *There exist distributional languages (A, μ) and (B, ν) such that $(A, \mu_A) \leq^p_T (B, \mu_B)$ but $(A, \mu_A) \not\leq^p_{tt} (B, \mu_B)$.*

Proof. We adapt the construction of Ladner Lynch and Selman [54]. A contains strings of the form 0^n where $n \in range(tower)$. We maintain the following relation

$$0^n \in A \Leftrightarrow \exists y, |y| = n, \forall y_1 \text{ where } y_1 \text{ is a prefix of } y, \langle 0^n, y_1 \rangle \in B. \qquad (6.4)$$

Let $T_1, T_2, \cdots$ be an enumeration of truth-table reducibilities such that T_k runs in time n^k. Let $Q(T_k(x))$ denote the queries produced by T_k on x.

Stage k: We diagonalize against the T_k. Let $n = tower(k)$. Find a y of length n such that $y \notin Q(T_k(0^n))$. Such a y always exists, because $|Q(T_k(0^n))| \leq n^k$. For every proper prefix y_1 of y place the tuple $\langle 0^n, y_1\rangle$ into B. If $x \in Q(T_k(0^n))$ and B is not defined on x yet, then place x into $\overline{B}$. Now, evaluate the truth table of T_k. If $T_k(0^n) = 1$, then place 0^n in $\overline{A}$ and $\langle 0^n, y\rangle$ in $\overline{B}$. If $T_k(0^n) = 0$, then place 0^n into A and $\langle 0^n, y\rangle$ into B.

It is clear from the construction that $A \not\leq_{tt}^p B$. Condition 6.4 ensures that $A \leq_T^p B$. Now we define the distributions μ and ν:

$$\mu'_A(x) = \frac{1}{n^2 2^n}$$

and

$$\mu'_B(\langle 0^n, y\rangle) = \frac{1}{n^2 2^{n+1}} \text{ if } n \in range(tower) \text{ and } |y| \leq n.$$

For all other strings x, $\mu'_B(x) = \frac{1}{n^2 2^n}$.

Now we show that $(A, \mu) \leq_T^p (B, \nu)$. If x is not a tally string, then x does not belong to A. If $x = 0^n$ and $n \in range(tower)$, then the reduction verifies whether Condition 6.4 is satisfied. Thus the reduction queries strings of the form $\langle 0^n, y\rangle$, where $|y| \leq n$. $\mu'(0^n) = 1/n^2 2^n$ and $\mu'_B(\langle 0^n, y\rangle) = 1/n^2 2^{n+1}$. Thus the domination condition is satisfied. □

We have not been able to separate Turing completeness from truth-table completeness for DistEXP. In the proof of Theorem 6.6, we have complete freedom to choose the distribution μ_A. This is crucial to the proof. Whereas to separate Turing completeness from truth-table completeness, we would have to exhibit a distributional problem (B, μ_B) such that (K_E, μ_{K_E}) is Turing reducible to (B, μ_B). Thus we would not have freedom to pick a distribution on K_E, we would have to work with μ_{K_E}. This seems to be a stumbling block.

Our next theorem shows another difficulty in separating Turing completeness from truth-table completeness for DistEXP. We show that if (B, μ_B) is Turing complete for DistEXP, then every tally language in EXP is truth-table reducible to B. In worst-case complexity, results that separate truth-table completeness from Turing completeness achieve the separation using a tally language [18, 54]. They construct a Turing complete language L and a tally language T and show that T is not truth-table reducible to L. The next theorem shows that this technique cannot be used to separate Turing completeness from truth-table completeness for DistEXP.

Theorem 6.7. *Let* (B, μ_B) *be a Turing complete problem for* DistEXP*. Let* (T, μ) *be a any distributional problem in* DistEXP*, where* T *is a tally language and* μ *is a polynomial-time computable distribution. Then* (T, μ) *is truth-table reducible to* (B, μ_B)*.*

Proof. Let ν be the following polynomial-time computable distribution.

$$\nu'(x) = \begin{cases} 1/2 & \text{if } x = 1, \\ 1/|x|^2 & \text{if } x = 0^n, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that (T, μ) is many-one reducible to (T, ν). Define f as follows: $f(x) = x$ if $x = 0^n$ and $f(x) = 1$ otherwise. We show that (T, ν) is truth-table reducible to (B, μ_B). Let M be the machine that witnesses the Turing reduction from (T, ν) to (B, μ_B). Since the reduction should satisfy the domination condition, there exists a polynomial p such that for every x, if M queries y on input x, then $\nu'(x) \leq p(|x|)\mu'_B(y)$. Thus $\mu'_B(y) \geq \nu'(x)/p(|x|)$.

We now describe the truth-table reduction. Since the reduction is obvious on non-tally strings we concentrate only on tally strings. On input 0^n the machine M can query only the strings whose weight is at least $1/n^2p(n)$. Note that there are at most $n^2p(n)$ many such strings. The truth-table reduction tries to generate all possible strings whose weight is at least $1/n^2p(n)$. At any stage of the computation the reduction may branch into two parallel computations. We will ensure that at any point of time there are at most polynomially many parallel branches. The procedure $\mathcal{A}$ in Figure 6.1 generates the queries.

It is clear that procedure $\mathcal{A}$ runs in polynomial time and Q contains all possible queries that M can make on 0^n. Given $\mathcal{A}$, the truth-table reduction is obvious.

Since (T, μ) is many-one reducible to (T, ν) and (T, ν) is truth-table reducible to (B, μ_B), (T, μ) is truth-table reducible to (B, μ_B). $\square$

Finally, Aida *et al.* [1] recently showed that if P $\neq$ NP, then truth-table reducibility can be separated from many-one reducibility in DistNP.

Theorem 6.8. ([1]) *If* P $\neq$ NP*, then there exist distributional problems* (A, μ_A) *and* (A, μ_B) *in* DistNP *such that* (A, μ_A) *is truth-table reducible to* (B, μ_B) *but is not many-one reducible.*

This result is very interesting because we do not know how separate different notions of reductions in NP under the hypothesis P $\neq$ NP.

input 0^n;
$Q = \phi$;
simulate M on 0^n;
  if M produces a query q
    then if $\mu'_B(q) < 1/n^2 p(n)$ or $q \in Q$
      then terminate the branch
      else
        $Q = Q \cup \{q\}$;
        Branch into two parallel computations;
        First branch proceeds with the answer "yes";
        Second branch proceeds with the answer "no"

Figure 6.1: Procedure $\mathcal{A}$

Chapter 7

Separation of NP-completeness Notions

In this chapter we study reductions in the context of worst-case complexity theory. Ladner, Lynch, and Selman [54] raised the natural question of whether various notions of reducibilities differ on NP. If there exist sets A and B in NP (other than the empty set or Σ^*) such that $A \leq_T^p B$ but $A \not\leq_m^p B$, then, of course, P $\neq$ NP follows immediately. With this in mind, they conjectured that P $\neq$ NP implies that $\leq_T^p$ and $\leq_m^p$ differ on NP.

In the intervening years, many results have explained the behavior of polynomial-time reducibilities within other complexity classes and have led to a complete understanding of the completeness notions that these reducibilities induce. For example, Ko and Moore [53] demonstrated the existence of $\leq_T^p$-complete sets for EXP that are not $\leq_m^p$-complete. Watanabe [90] extended this result significantly, showing that $\leq_{1-tt}^p$-, $\leq_{btt}^p$-, $\leq_{tt}^p$-, and $\leq_T^p$-completeness for EXP are mutually different, while Homer, Kurtz, and Royer [42] proved that $\leq_m^p$- and $\leq_{1-tt}^p$-completeness are identical.

However, there have been few results comparing reducibilities within NP, and we have known very little concerning various notions of NP-completeness.

It is surprising that no NP-complete problem has been discovered that requires anything other than many-one reducibility for proving its completeness. The first result to distinguish reducibilities within NP is an observation of Wilson in one of Selman's papers on p-selective sets [78]. It is a corollary of results there that if NE $\cap$ *co*-NE $\neq$ E, then there exist sets A and B belonging to NP such that $A \leq_{postt}^p B$, $B \leq_{tt}^p A$, and $B \not\leq_{postt}^p A$, where $\leq_{postt}^p$ denotes positive truth-table reducibility. Regarding completeness, Longpré and Young [58] proved that there are $\leq_m^p$-complete sets for NP for which $\leq_T^p$-reductions to these sets are *faster*, but they did not prove that the completeness notions differ. The first to give technical evidence that $\leq_T^p$- and $\leq_m^p$-completeness for NP differ were Lutz and Mayordomo [62], who proved that if the p-measure of NP is not zero, then there exists a $\leq_{3\text{-}tt}^p$-complete set that is not $\leq_m^p$-complete. Ambos-Spies and Bentzien [4] extended this result significantly. They used an hypothesis of resource-bounded category theory that is weaker than that of Lutz and Mayordomo to separate nearly all NP-completeness notions for the bounded truth-table reducibilities.

It has remained an open question as to whether we can separate NP-completeness notions without using hypotheses that involve essentially stochastic concepts. Furthermore, the only comparisons of reducibilities within NP known to date have been those just listed.

Here we report some new progress on these questions. Our main new result introduces a strong, but reasonable, hypothesis to prove the existence of a $\leq_T^p$-complete set in NP that is not $\leq_{tt}^p$-complete. Our hypothesis is similar to, but seemingly stronger than, hypotheses considered by researchers previously, notably Fenner, Fortnow, Naik, and Rogers [22], Hemaspaandra, Rothe and Wechsung [41], and Fortnow, Pavan, and Selman [25].

This result is especially interesting because the measure theory and category theory techniques seem to be successful primarily for the nonadaptive reducibilities. We also show that a weaker version of our hypothesis separates Turing completeness from many-one completeness. We first present the separation results then we provide a fairly thorough analyses of our hypotheses.

7.1 Separation Results

Let Hypothesis H be the following assertion:

Hypothesis H: There is a UP-machine M that accepts 0^* such that

1. no polynomial time-bounded Turing machine correctly computes infinitely many accepting computations of M, and

2. for some $\epsilon > 0$, no 2^{n^ϵ} time-bounded Turing machine correctly computes all accepting computations of M.

Theorem 7.1. *If Hypothesis H is true, then there exists a $\leq_T^p$-complete language for* NP *that is not $\leq_{tt}^p$-complete for* NP.

Proof. Let M be a UP-machine that satisfies the conditions of Hypothesis H. For each $n \geq 0$, let a_n be the unique accepting computation of M on 0^n, and let $l_n = |a_n|$. Define the language

$$L_1 = \{\langle x, a_n \rangle \mid |x| = n, \text{ and } x \in \text{SAT}\}.$$

Define the infinite string $a = a_1 a_2 \ldots$, and define

$$L_2 = L(a) = \{x \mid x < a\}$$

to be the standard left-cut of a.

We define $L = L_1 \oplus L_2$ to be the disjoint union of L_1 and L_2. We will prove that L is $\leq_T^p$-complete for NP but not $\leq_{tt}^p$-complete.

Lemma 7.1. *L is $\leq_T^p$-complete for* NP.

Proof. It is clear that L belongs to NP. The following reduction witnesses that SAT $\leq_T^p L$: Given an input string x, where $|x| = n$, use a binary search algorithm that queries L_2 to find a_n. Then, note that $x \in \text{SAT}$ if and only if $\langle x, a_n \rangle$ belongs to L_1. □

Lemma 7.2. *L is not $\leq_{tt}^p$-complete for* NP.

Proof. Assume that L is $\leq_{tt}^p$-complete for NP. Define the set

$$S = \{\langle 0^n, i \rangle \mid \text{ the } i\text{-th bit of } a_n = 1\}.$$

Clearly, S belongs to NP. Thus, by our assumption, there is a $\leq_{tt}^p$-reduction $\langle g, h \rangle$ from S to L. Given this reduction, we will derive a contradiction to Hypothesis H.

Consider the following procedure $\mathcal{A}$:

1. input 0^n;
2. compute the sets $Q^j = g(\langle 0^n, j \rangle)$, for $1 \leq j \leq l_n$. Let $Q = \bigcup\{Q^j \mid 1 \leq j \leq l_n\}$;
3. Let Q_1 be the set of all queries in Q to L_1 and let Q_2 be the set of all queries in Q to L_2 ($Q = Q_1 \cup Q_2$);
4. If Q_1 contains a query $\langle x, a_t \rangle$, where $t \geq n^\epsilon$, then output "Unsuccessful" and Print a_t, else output "Successful."

Observe that this procedure runs in polynomial time. We treat two cases, namely, either $\mathcal{A}(0^n)$ is unsuccessful, for infinitely many n, or it is successful, for all but finitely many n.

Claim 7.1. *If the procedure $\mathcal{A}(0^n)$ is unsuccessful for infinitely many n, then there is a polynomial time-bounded Turing machine that correctly computes infinitely many accepting computations of M, thereby contradicting Clause 1 of Hypothesis H.*

Proof. If $\mathcal{A}(0^n)$ is unsuccessful, then it outputs a string a_t such that $t \geq n^\epsilon$. Hence, if $\mathcal{A}(0^n)$ is unsuccessful for infinitely many n, then for infinitely many t there exists an n, where $n \leq t^{1/\epsilon}$, and $\mathcal{A}(0^n)$ outputs a_t. The following procedure uses this observation to compute infinitely many accepting computations of M in polynomial time.

input 0^t;
for $j = 1$ to $j = t^{1/\epsilon}$ **do**
  if $\mathcal{A}(0^j)$ outputs a_t
    then output a_t and halt.

The procedure runs in polynomial time because the procedure $\mathcal{A}(0^j)$ runs in polynomial time. □

Claim 7.2. *If $\mathcal{A}(0^n)$ is successful for all but finitely many n, then there is a 2^{n^ϵ} time-bounded Turing machine that correctly computes all accepting computations of M, thereby contradicting Clause 2 of Hypothesis H.*

Proof. We will demonstrate a procedure $\mathcal{B}$ such that for each n, if $\mathcal{A}(0^n)$ is successful, then $\mathcal{B}$ on input 0^n outputs the accepting computation of M on 0^n in 2^{n^ϵ} time.

If $\mathcal{A}(0^n)$ is successful, then no member of the set Q_1 is of the form $\langle x, a_t \rangle$ where $t \geq n^\epsilon$. We begin our task with the following procedure $\mathcal{C}$ that for each query $q = \langle y, z \rangle$ in Q_1 decides whether $q \in L_1$.

1. input $q = \langle y, z \rangle$;
2. If $z \neq a_t$ for some t, then $\langle y, z \rangle$ does not belong to L_1; (This can be determined in polynomial time.)
3. if $z = a_t$, where $t \leq n^\epsilon$, then $\langle y, z \rangle$ belongs to L_1 only if $|y| = t$ and y belongs to SAT. (Since $t \leq n^\epsilon$ this step can be done in time 2^{n^ϵ}).

Thus, $\mathcal{C}$ decides membership in L_1 for all queries q in Q_1. Therefore, if for each query q in Q_2, we can decide whether q belongs to L_2, then the evaluator h can determine whether each input $\langle 0^n, j\rangle$, $1 \le j \le l_n$, belongs to S. That is, if for each query q in Q_2, we can decide whether q belongs to L_2, then we can compute a_n. We can accomplish this using a standard proof technique for p-selective sets [39, 82]. Namely, since L_2 is a standard left-cut, by Lemma 2.1, there exists a pivot string z in $Q_2 \cup \{\perp\}$ such that $Q_2 \cap L_2$ is the set of all strings in Q_2 that are less than or equal to z. We do not know which string is the pivot string, but there are only $\|Q_2\|$ choices, which is a polynomial number of choices. Thus, procedure $\mathcal{B}$ on input 0^n proceeds as follows to compute a_n: For each possible choice of pivot and the output from procedure $\mathcal{C}$, the evaluator h computes a possible value for each j-th bit of a_n. There are only a polynomial number of possible choices of a_n, because there are only a polynomial number of pivots. $\mathcal{B}$ verifies which choice is the correct accepting computation of M on 0^n, and outputs that value. Finally, we have only to note that the entire process can be carried out in 2^{n^ϵ} steps. □

Lemma 7.2 follows from Claims 7.1 and 7.2. □

The theorem follows from Lemmas 7.1 and 7.2. □

Let Hypothesis H' be the following assertion:

Hypothesis H'**:** There is an NP-machine M that accepts 0^* such that for some $0 < \epsilon < 1$, no 2^{n^ϵ} time-bounded Turing machine correctly computes infinitely-many accepting computations of M.

Theorem 7.2. *If Hypothesis* H' *is true, then there exists a Turing complete language for* NP *that is not* $\le_m^p$*-complete for* NP*.*

Proof. Let M be an NP-machine that satisfies the conditions of Hypothesis H'. For each $n \ge 0$, let a_n be the lexicographically maximum accepting computation of M on 0^n, and let $|a_n| = l_n$. Define the language

$$L_1 = \{\langle x, u\rangle \mid |x| = n,\ u \text{ is an accepting computation of } M \text{ on } 0^m, n = m^\epsilon/2, \text{ and } x \in \mathrm{SAT}\}.$$

Let $a = a_1 a_2 a_3 \cdots$, and define

$$L_2 = L(a) = \{x \mid x < a\}.$$

Define $L = L_1 \oplus L_2$.

It is easy to see, as in the previous argument, that L is $\leq_T^p$-complete for NP. In order to prove that L is not $\leq_m^p$-complete, we define the set

$$S = \{\langle 0^n, y\rangle \mid y \text{ is a prefix of an accepting computation of } M \text{ on } 0^n\},$$

which belongs to NP, and assume there is a $\leq_m^p$-reduction f from S to L. Consider the procedure $\mathcal{D}$ in Figure 7.1: First we will analyze the running time and then we treat two cases, namely, either $\mathcal{D}(0^n)$ is successful for infinitely many n, or it is unsuccessful for all but finitely many n.

Claim 7.3. *The above procedure halts in* $O(l_n 2^{n^{\epsilon^2}/2})$ *steps.*

Proof. Consider an iteration of the repeat loop. The most expensive step is the test of whether "$z \in$ SAT". This test occurs only when $|z| = t^\epsilon/2$ and $t < n^\epsilon$. Hence we can decide whether z belongs to SAT in $2^{n^{\epsilon^2}/2}$ steps. All other steps take polynomial time. Hence the time taken by the procedure is $O(l_n 2^{n^{\epsilon^2}/2})$. □

Since $0 < \epsilon < 1$, the running time of procedure $\mathcal{D}$ is bounded by 2^{n^ϵ}.

Claim 7.4. *If* $\mathcal{D}(0^n)$ *is successful for infinitely many* n*, then there is a* 2^{n^ϵ}*-time-bounded Turing machine that correctly computes infinitely many accepting computations of* M*.*

Proof. We demonstrate that if $\mathcal{D}$ is successful on an input 0^n, then the string that is printed is an accepting computation of M on 0^n. In order to accomplish this, we prove by induction that y is a prefix of an accepting computation of M on 0^n during every iteration of the repeat loop (i.e., a loop invariant). Initially when $y = \lambda$ this is true. Assume that y is a prefix of an accepting computation of M at the beginning of an iteration. Then, at least one of $f(\langle 0^n, y0\rangle) = x_0$, $f(\langle 0^n, y1\rangle) = x_1$ must belong to L. If both x_0 and x_1 are queries to L_2, then the smaller of x_0 and x_1 belongs to L_2 because L_2 is p-selective. Thus, in this case, the procedure extends y correctly. If at least one of x_0 and x_1 is a query to L_1, then the procedure determines whether $x_b \in L_1$, where x_b is the query to L_1 with least index. If x_b belongs to L, then $\langle 0^n, yb\rangle \in S$. Hence, yb is a prefix of an accepting computation. If $x_b \notin L$, then $x_{\bar{b}}$ belongs to L, because at least one of x_b or $x_{\bar{b}}$ belongs to L. Thus, in this case, $y\bar{b}$ is a prefix of an accepting computation. This completes the induction argument.

```
input 0^n;
y := λ;
Repeat l_n times
    begin
    f(⟨0^n, y0⟩) := x_0;
    f(⟨0^n, y1⟩) := x_1;
    if both x_0 and x_1 are queries to L_2
        then if x_0 ≤ x_1
            then y := y0
            else y := y1
        else {At least one of x_0 and x_1 is a query to L_1;
                let b ∈ {0, 1} be the least index
                such that x_b queries L_1, and let x_b = ⟨z, u⟩.}
            if u is not an accepting computation of M {thus, x_b ∉ L_1}
                then y = y b̄
                else {u is an accepting computation of M on 0^t}
                    if t ≥ n^ε
                        then output "Unsuccessful," print u, and terminate
                        else {t < n^ε}
                            if |z| = t^ε/2 and z ∈ SAT {thus, x_b ∈ L_1}
                                then y := yb
                                else {x_b ∉ L_1} y := y b̄
    end;
output "Successful" and print y.
```

Figure 7.1: Procedure $\mathcal{D}$

The loop repeats l_n times. Therefore, the final value of y, which is the string that $\mathcal{D}$ prints, is an accepting computation. $\square$

Claim 7.5. *If $\mathcal{D}(0^n)$ is unsuccessful for all but finitely many n, then there is a 2^{n^ϵ}-time-bounded Turing machine that correctly computes infinitely many accepting computations of M.*

Proof. The proof is similar to the proof of Claim 7.1. The following procedure computes infinitely many accepting computations of M.

input 0^n;
for $j = 1$ to $j = n^{1/\epsilon}$ **do**
  if $\mathcal{D}(0^j)$ outputs u and u is an accepting computation of M on 0^n
    then print u and terminate.

The running time of this algorithm can be bounded as follows: The procedure $\mathcal{D}(0^j)$ runs in time $l_j 2^{j^{\epsilon^2}/2}$ steps. So the total running time is $\sum_1^{n^{1/\epsilon}} l_j 2^{j^{\epsilon^2}/2} = O(2^{n^\epsilon})$. $\square$

Since the cases treated both by Claims 7.4 and 7.5 demonstrate Turing machines that correctly compute infinitely many accepting computations of M in 2^{n^ϵ} time, we have a contradiction to Hypothesis H$'$. Thus L is not $\leq_m^p$-complete for NP. $\square$

Next we note some extensions. These extensions are independenty observed by Regan and Watanabe [71]. In Hypothesis H we can replace the UP-machine by an NP-machine under a stronger intractability assumption. Consider the following hypothesis:

There is a NP-machine M that accepts 0^* such that

1. no probabilistic polynomial time-bounded Turing machine correctly outputs infinitely many accepting computations with non-trivial (inverse polynomial) probability, and
2. for some $\epsilon > 0$, no 2^{n^ϵ} time-bounded Turing machine correctly computes all accepting computations with non-trivial probability.

We can prove that Turing completeness is different from truth-table completeness in NP under the above hypothesis. The proof uses the randomized reduction of Valiant and Vazirani [84] that isolates the accepting computations. We define L as in the proof of Theorem 7.2. Let

$$\begin{aligned} S &= \{\langle 0^n, k, r_1, r_2, \cdots, r_k, i\rangle \mid \exists v \text{ such that } v \text{ is an accepting computation} \\ &\quad \text{of } M,\ v.r_1 = v.r_2 = \cdots = v.r_k = 0, \text{ and the } i\text{th bit of } v = 1\} \end{aligned}$$

where $v.r_i$ denotes the inner product over GF[2].

Valiant and Vazirani showed that if we randomly pick $r_1, r_2, \cdots, r_k$, then with a non-trivial probability there exists exactly one accepting computation v of M whose inner product with each r_i is 0. Thus, for a random choice of $r_1, \cdots, r_k$, there is exactly one witness v for $\langle 0^n, k, r_1, \cdots, r_k, i\rangle$. The rest of the proof is similar to that of Theorem 7.1.

We also note that we can replace the UP-machine in Hypothesis H with a FewP-machine.

Now we turn our attention to separating reducibilities in NP. The following results give fine separations of polynomial time reducibilities in NP from significantly weaker hypotheses. Moreover, they follow readily from results in the literature.

Theorem 7.3. *If there is a tally language in* UP $-$ P*, then there exist two languages* L_1 *and* L_2 *in* NP *such that* $L_1 \leq_{tt}^p L_2$*,* $L_2 \leq_T^p L_1$*, but* $L_1 \not\leq_{btt}^p L_2$.[1]

Proof. Let L be a tally language in UP $-$ P. Let R be the polynomial-time computable relation associated with the language L. Define

$$L_1 = \{\langle 0^n, y\rangle \mid \exists w, R(0^n, w) \text{ and } y \leq w\}$$

and

$$L_2 = \{\langle 0^n, i\rangle \mid \exists w, R(0^n, w) \text{ and } i\text{-th bit of } w \text{ is one}\}.$$

It is clear that L_1 is $\leq_{tt}^p$-reducible to L_2. To see that L_2 is $\leq_T^p$-reducible to L_1, implement a binary search algorithm that accesses L_1 to determine the unique witness w such that $R(0^n, w)$, and then find the i-th bit.

Observe that L_2 is a sparse set. Ogiwara and Watanabe [65] called L_1 the *left set* of L, and they and Homer and Longpré [43] proved for every L in NP that if the left set of L is $\leq_{btt}^p$-reducible to a sparse set, then L is in P. Hence $L_1 \not\leq_{btt}^p L_2$. □

[1]The class of all languages that are $\leq_T^p$-equivalent to L_1 is a noncollapsing degree.

We now prove that Turing and truth-table reducibilities also differ in NP under the same hypothesis.

Theorem 7.4. *If there is a tally language in* UP − P*, then there exist two languages* L_1 *and* L_2 *in* NP *such that* $L_1 \leq_T^p L_2$ *but* $L_1 \not\leq_{tt}^p L_2$*.*

Proof. Hemaspaandra et al. [39] proved that the hypothesis implies existence of a tally language L in UP − P such that L is not $\leq_{tt}^p$-reducible to any p-selective set. In the same paper they also showed, given a tally language L in NP − P, how to obtain a p-selective set S such that L is $\leq_T^p$-reducible to S. Combing the two results we obtain the theorem. □

7.2 Analysis of the Hypotheses

This section contains a number of results that help us to understand the strength of Hypotheses H and H$'$.

7.2.1 Comparisons with Other Complexity-Theoretic Assertions

We begin with some equivalent formulations of these hypotheses, and then relate them to other complexity-theoretic assertions. The question of whether P contains a P-printable-immune set was studied by Allender and Rubinstein [2], and the equivalence of items 1 and 3 in the following theorem is similar to results of Hemaspaandra, Rothe, and Wechsung [41] and Fortnow, Pavan, and Selman [25]. The second item is similar to the the characterization of Grollmann and Selman [30] of *one-one, one-way* functions with the addition of the attribute *almost-always one-way* of Fortnow, Pavan, and Selman.

Theorem 7.5. *The following statements are equivalent:*

1. *There is a language* L *in* P *that contains exactly one string of every length such that* L *is* P*-printable-immune and, for some* $\epsilon > 0$*,* L *is not* 2^{n^ϵ}*-printable.*

2. *There exists a polynomial-bounded, one-one, function* $f : 0^* \to \Sigma^*$*, such that* f *is almost-everywhere not computable in polynomial time, for some* $\epsilon > 0$*,* f *is not computable in time* 2^{n^ϵ}*, and the graph of* f *belongs to* P*.*

3. *Hypothesis H is true for some $\epsilon > 0$.*

Proof. Let L satisfy item one. Define

$$f(0^n) = \text{ the unique string of length } n \text{ that belongs to } L.$$

Clearly, f us polynomial-bounded and one-one. The graph of f belongs to P, because L belongs to P. Suppose that M is a Turing machine that computes f and that runs in polynomial time on infinitely many inputs. Then, on these inputs, M prints $L \cap \Sigma^n$. Similarly, f is not computable in time 2^{n^ϵ}.

Let f satisfy item two. Define a UP-machine M to accept 0^* as follows: On input 0^n, M guesses a string y of length within the polynomial-bound of f, and accepts if and only if $\langle 0^n, y \rangle \in graph(f)$. The rest of the proof is clear.

Let M be a UP-machine that satisfies item three, i.e., that satisfies the conditions of Hypothesis H. Let a_n be the unique accepting computation of M on 0^n and let $|a_n| = n^l$. Let r_n be the rank of a_n among all strings of length n^l. Now, we define L as follows: Given a string x, if $|x| = n^l$ for some n, then x belongs to L if and only if $x = a_n$. If $(n-1)^l < |x| < n^l$, then x belongs to L if and only if the rank of x (among all the string of length $|x|$) is r_{n-1}. It is clear that $L \in$ P and has exactly one string per each length. We claim that L is P-printable-immune and is not 2^{n^ρ}-printable, where $\epsilon = l\rho$. Any machine that prints infinitely many strings of L in polynomial time can be used to print infinitely many accepting computations of M in polynomial time. Thus L is P-printable-immune. Any machine that prints all the strings of L in 2^{n^ρ} time can be used print all the accepting computations of M in 2^{n^ϵ} time. Thus L is not 2^{n^ρ}-printable. □

We prove the following theorem similarly.

Theorem 7.6. *The following statements are equivalent.*

1. *There is a language L in* P *that contains* at least *one string of every length such that, for some $\epsilon > 0$, L is 2^{n^ϵ}-printable-immune.*

2. *There is polynomial-bounded, multivalued function $f : 0^* \to \Sigma^*$ such that every refinement of f is almost-everywhere not computable in 2^{n^ϵ}-time, and the graph of f belongs to* P.

3. *Hypothesis $H^{'}$ holds for some $\epsilon > 0$.*

Next we compare our hypotheses with the following complexity-theoretic assertions:

1. For some $\epsilon > 0$, there is a P-bi-immune language L in $\mathrm{UP} \cap \text{co-UP}$ such that $L \cap 0^*$ is not in $\mathrm{DTIME}(2^{n^\epsilon})$.
2. For some $\epsilon > 0$, there is language L in $\mathrm{UP} \cap \text{co-UP}$ such that L is not in $\mathrm{DTIME}(2^{n^\epsilon})$.
3. For some $\epsilon > 0$, there is a 2^{n^ϵ}-bi-immune language in $\mathrm{NP} \cap \text{co-NP}$.

Theorem 7.7. *Assertion 1 implies Hypothesis H and Hypothesis H implies Assertion 2.*

Proof. Let L be a language in $\mathrm{UP} \cap \text{co-UP}$ that satisfies Assertion 1. Define M to be the UP-machine that accepts 0^* as follows: On input 0^n, nondeterministically guess a string w. If w either witnesses that 0^n is in L or witnesses that 0^n is in $\overline{L}$, then accept 0^n. It is immediate that M satisfies the conditions of Hypothesis H.

To prove the second implication, let M a UP-machine that satisfies the conditions of Hypothesis H. Let a_n denote the unique accepting computation of M on 0^n and define

$$L = \{\langle 0^n, x\rangle \mid x \leq a_n\}.$$

It is clear that $L \in \mathrm{UP} \cap \text{co-UP}$. If $L \in \mathrm{DTIME}(2^{n^\epsilon})$, then a binary search algorithm can correctly compute a_n, for every n, in time 2^{n^ϵ}. This would contradict Hypothesis H. Hence, $L \notin \mathrm{DTIME}(2^{n^\epsilon})$. □

The discrete logarithm problem is an interesting possible witness for Assertion 2. The best known deterministic algorithm requires time greater than $2^{n^{\frac{1}{3}}}$ [29]. Thus, the discrete logarithm problem is a candidate witness for the noninclusion $\mathrm{UP} \cap \textit{co}\text{-UP} \not\subseteq \mathrm{DTIME}(2^{n^\epsilon})$, for any $0 < \epsilon \leq \frac{1}{3}$.

Corollary 7.1. *If, for some $\epsilon > 0$, $\mathrm{UP} \cap \text{co-UP}$ has a 2^{n^ϵ}-bi-immune language, then $\leq_T^p$-completeness is different from $\leq_{tt}^p$-completeness for* NP.

Theorem 7.8. *Assertion (3) implies Hypothesis H'.*

Corollary 7.2. *If, for some $\epsilon > 0$, $\mathrm{NP} \cap \text{co-NP}$ has a 2^{n^ϵ}-bi-immune language, then $\leq_T^p$-completeness is different from $\leq_m^p$-completeness for* NP.

7.2.2 Comparisons with Genericity

The genericity hypothesis of Ambos-Spies and Bentzien [4], which they used successfully to separate NP-completeness notions for the bounded-truth-table reducibilities, states that "NP contains an n^2-generic language." Our next result enables us to compare this with our hypotheses.

We say that a deterministic oracle Turing machine M is a *predictor* for a language L if for every input word x, M decides whether $x \in L$ with oracle $L|x$. L is *predictable in time* $t(n)$ if there is a $t(n)$ time-bounded predictor for L. We define a set L to be *almost-everywhere unpredictable in time* $t(n)$ if every predictor for L requires more than $t(n)$ time for all but finitely many x. This concept obviously implies DTIME($t(n)$)-complex almost everywhere, but the converse does not hold:

Theorem 7.9. EXP *contains languages that are* DTIME(2^n)*-complex but* not *almost-everywhere unpredictable in time* 2^n*.*

Now we state our characterization of $t(n)$-genericity.

Theorem 7.10. *Let $t(n)$ be a polynomial. A decidable language L is $t(n)$-generic if and only if it is almost-everywhere unpredictable in time $t(2^n - 1)$.*

Proof. Assume that L is not almost-everywhere unpredictable in time $t(2^n - 1)$, and let M be a predictor for L that for infinitely many strings x runs in time $t(2^n - 1)$. Define a condition C so that the characteristic sequence

$$(L|x)\overline{x} \in C \Leftrightarrow M \text{ with oracle } L|x \text{ runs in time } t(2^{|x|} - 1) \text{ on input } x,$$

where $\overline{x} = \neg(M \text{ accepts } x)$. Then, C is dense along L because M correctly predicts whether $x \in L$ for infinitely many x. It is easy to see that $C \in \mathrm{DTIME}(t(n))$. However, L is not $t(n)$-generic because we defined C so that L does not meet C.

Assume that L is not $t(n)$-generic, and let $C \in \mathrm{DTIME}(t(n))$ be a condition that is dense along L such that L does not meet C. Let T be a deterministic Turing machine that halts on all inputs and accepts L. Define a predictor M for L to behave as follows on input x with oracle $A|x$: If $(A|x)1 \in C$, then M rejects x, and if $(A|x)0 \in C$, then M accepts x. If neither holds, then M determines membership in L by simulating T on x. Since L does not meet C, M is a predictor for L. Since C is dense along L and L does not meet C, for infinitely many x, either $(A|x)1 \in C$ or $(A|x)0 \in C$, and in each of these cases, M runs for at most $t(2 \cdot 2^{|x|})$ steps. Since $t(n)$ is polynomial function, by the linear speedup theorem [36], there is a Turing machine that is equivalent to M that runs in time $t(2^{|x|} - 1)$. □

Corollary 7.3. NP *contains an* n^2*-generic language if and only if* NP *contains a set that is almost-everywhere unpredictable in time* 2^{2n}.

By Theorem 7.8, Hypothesis H$'$ holds if NP∩co-NP contains a set that, for some $\epsilon > 0$, is 2^{n^ϵ}-bi-immune. So, Hypothesis H$'$ requires bi-immunity, which is weaker than almost-everywhere unpredictability, and the time-bound is reduced from 2^{2n} to 2^{n^ϵ}. On the other hand, we require the language to belong to NP ∩ co-NP instead of NP. Similarly, when we consider Hypothesis H, we require the language to be P-bi-immune and not in DTIME(2^{n^ϵ}), whereas now we require the language to be in UP ∩ co-UP. Moreover, the conclusion of Theorem 7.1 is not known to follow from the genericity hypothesis. At the same time, we note that the genericity hypothesis separates several bounded-truth-table completeness notions in NP that do not seem obtainable from our hypotheses.

7.2.3 Relativization

Theorem 7.11. *There exists an oracle relative to which the polynomial hierarchy is infinite and Hypotheses H and* H$'$ *both hold.*

Proof. Define Kolmogorov random strings $r_0, r_1, \ldots$ as follows: r_n is the first string of length n such that

$$K^{2^n}(r_n \mid r_0, r_1, \ldots, r_{n-1}) > n/2.$$

Then, define the oracle $A = \{r_n \mid n \geq 0\}$.

Define M to be an oracle Turing machine that accept 0^* with oracle A as follows: On input 0^n, guess a string y of length n. If $y \in A$, then accept. M is a UP^A-machine that accepts 0^* because A contains exactly one string of every length.

Now we show that no 2^{n^ϵ} oracle Turing machine with oracle A, for any $0 < \epsilon < 1$, correctly computes infinitely many accepting computations of M. Observe that relative to A, this implies both Hypotheses H and H$'$. Suppose otherwise, and let T be such an oracle Turing machine. The gist of the remainder of the proof is that we will show how to simulate T without using the oracle, and that will contradict the randomness or r_n.

Suppose that $T^A(0^n) = r_n$. Let $l = 3n^\epsilon$. Then we simulate this computation without using an oracle as follows:

1. Compute $r_0, r_1, \ldots, r_{l-1}$. Do this iteratively: Compute r_i by running every program (with input strings $r_0, r_1, \ldots, r_{i-1}$) of length $\leq i/2$ for 2^i steps. Then r_i is the first string of length i that is not output by any of these programs. Note that the total time for executing this step is

$$l 2^{l/2} 2^l \leq l 2^{3l/2} \leq 2^{5n^\epsilon}.$$

2. Simulate T on input 0^n, except replace all oracle queries q by the following rules: If $|q| < l$, answer using the previous computations. Otherwise, just answer "no."

If the simulation is correct, then this procedure outputs r_n without using the oracle. The running time of this procedure on input 0^n is $2^{5n^\epsilon} + 2^{n^\epsilon}$, which is less than 2^n. So, we can describe r_n by a string of length $O(\log n)$, to wit, a description of T and 0^n. This contradicts the definition of r_n.

We need to show that the simulation is correct. The simulation can only be incorrect if $|q| \geq l$ and $q = r_m$, for some $m > l$. Let r_m be the first such query. This yields a short description of r_m, given $r_0, r_1, \ldots, r_{l-1}$. Namely, the description consists of the description of T (a constant), the description of 0^n ($\log n$ bits), and the description of the number j such that $q = r_m$ is the j-th query (at most n^ϵ). Thus, the length of the description is $O(n^\epsilon)$. Since $l = 3n^\epsilon$, it follows that the length of the description of r_m is less than $m/2$. The running time of T, given $r_0, r_1, \ldots, r_{l-1}$, is 2^{n^ϵ}, which is less than 2^m. (The reason is that the first step in the simulation of T is not needed.) Therefore, the simulation is correct.

Note that A is a sparse oracle. Let B be an oracle relative to which the polynomial-time hierarchy is infinite [92, 38]. By applying the results of Balcázar *et al* [10] and Fortnow [23] to A and B, we obtain an oracle relative to which the polynomial-time hierarchy is infinite and Hypotheses H and H$'$ are true. □

Hypothesis H fails relative to any oracle for which P = NP $\cap$ co-NP [9]. Fortnow and Rogers [26] obtained an oracle relative to which NP $\neq$ co-NP and Hypothesis H$'$ fails.

We conclude this chapter by mentioning a few open problems. It would be interesting to find weaker hypotheses that separate completeness notions.

We do not know whether there exists an oracle relative to which $P \neq NP$ and every nontrivial[2] Turing complete set for NP is many-one complete.

[2] A set S is nontrivial if neither $S = \emptyset$ nor $S = \Sigma^*$.

Bibliography

[1] S. Aida, R. Schuler, T. Tsukiji, and O. Watanabe. On the difference between polynomial-time many-one and truth-table reducibilities on distributional problems. In *18th International Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 2010*, pages 51–62, Springer-Verlag, 2001.

[2] E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1988.

[3] K. Ambos-Spies. Resource-bounded genericity. In *Computability, Enumerability, Unsolvability*, volume 224 of *London Mathematical Society.* Cambridge University Press, 1996.

[4] K. Ambos-Spies and L. Bentzien. Separating NP-completeness under strong hypotheses. *Journal of Computer and System Sciences*, 61(3):335–361, 2000.

[5] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial time computable sets. *Theoretical Computer Science*, 51:177–204, 1987.

[6] K. Ambos-Spies and Mayordomo. Resource-bounded measure and randomness. In *Complexity, Logic, and Recursion Theory*, pages 1–47. Marcel Dekker, Inc, 1997.

[7] K. Ambos-Spies, H. Neis, and A. Terwijn. Genericity and measure for exponential time. *Theoretical Computer Science*, 168(1):3–19, 1996.

[8] K. Ambos-Spies, A. Terwijn, and X. Zheng. Resource bounded randomness and weakly complete problems. *Theoretical Computer Science*, 172(1):195–207, 1997.

[9] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM Journal on computing Computing*, 4(4):431–441, Dec. 1975.

[10] J. Balcázar, R. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of Association of Computing Machinery*, 33(3):603–617, 1986.

[11] J. Balcázar, J. Diaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, Berlin, 1988.

[12] J. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *Mathematical Systems Theory*, 18(1):1–18, June 1985.

[13] J. Belanger, A. Pavan, and J. Wang. Reductions do no preserve fast convergence rates in average time. *Algorithmica*, 23:363–373, 1999.

[14] J. Belanger and J. Wang. Reductions and convergence rates of average time. In *Proceedings of the Second Annual International Computing and Combinatorics Conference, Lecture Notes in Computer Science*, volume 1090, pages 300–309. Springer-Verlag, 1996.

[15] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992.

[16] A. Blass and Y. Gurevich. Matrix transformation is complete for the average case. *SIAM Journal of Computing*, 24:3–29, 1995.

[17] R. Book. Tally languages and complexity classes. *Information and Control*, 26:186–193, 1974.

[18] H. Buhrman, S. Homer, and L. Torenvliet. Completeness notions for nondeterministic complexity classes. *Mathematical Systems Theory*, 24:179–200, 1991.

[19] J. Cai and A. Selman. Fine separation of average time complexity classes. *SIAM Journal on Computing*, 28:1310–1325, 1999.

[20] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science*, pages 24–30. North Holland, 1964.

[21] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[22] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings of the 11th IEEE Conference on Computational Complexity*, pages 213–223, 1996.

[23] L. Fortnow. Relativized worlds with an infinite hierarchy. *Information Processing Letters*, 69(6):1310–1325, 1999.

[24] L. Fortnow, A. Pavan, and A. Selman. Distributionally-hard languages. In *COCOON 99, Lecture Notes in Computer Science 1627*, pages 184–193. Springer-Verlag, 1999.

[25] L. Fortnow, A. Pavan, and A. Selman. Distributionally-hard languages. *Theory of Computing Systems*, 34(3):245–262, 2001.

[26] L. Fortnow and J. Rogers. Separability and one-way functions. In D.Z. Du and X.S. Zhang, editors, *Proceedings of the Fifth International Symposium on Algorithms and Computation, Lecture Notes in Computer Science*, pages 396–404. Springer-Verlag, 1994.

[27] J. Geske, D. Huynh, and J. Seiferas. A note on almost-everywhere-complex sets and separating deterministic-time-complexity classes. *Information and Computation*, 92(1):97–104, 1991.

[28] A. Goldberg and M. Sipser. Compression and ranking. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 440–448, 1985.

[29] D. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM Journal on Discrete Mathematics*, 6:124–138, 1993.

[30] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–355, April 1988.

[31] Y. Gurevich. Matrix decomposition problem is complete for the average case. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 802–811, 1990.

[32] Y. Gurevich. Average case completeness. *Journal of Computer and System Sciences*, 42:346–398, 1991.

[33] Y. Gurevich and S. Shelah. Expected computation time for hamiltonian path problem. *SIAM Journal on Computing*, 16:486–502, 1987.

[34] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65:158–181, 1985.

[35] J. Hartmanis, P. Lewis, and R. Stearns. Hierarchies of memory limited computations. In *Proceedings of the Sixth Annual IEEE Symposium on Switching Circuit Theory and Logical Design*, pages 179–190, 1965.

[36] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[37] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34:17–32, 1984.

[38] J. Håstad. Almost optimal lower bounds for small depth circuits. In S. Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press, Greenwich, 1989.

[39] E. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. P-selective sets and reducing search to decision vs. self-reducibility. *Journal of Computer and System Sciences*, 53(2):194–209, 1996. Special Issue of papers selected from the Eighth Annual IEEE Conference on Structure in Complexity Theory. In press.

[40] L. Hemaspaandra and S. Jha. Defying upward and downward separation. *Information and Control*, 121(1):1–13, 1995.

[41] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34(11):859–879, 1997.

[42] S. Homer, S. Kurtz, and J. Royer. On 1-truth-table-hard languages. *Theoretical Computer Science*, 115(2):383–389, 1993.

[43] S. Homer and L. Longpré. On reductions of NP sets to sparse sets. *Journal of Computer and System Sciences*, 48(2):324–336, 1994.

[44] S. Homer and W. Maass. Oracle dependent properties of the lattice of NP sets. *Theoretical Computer Science*, 24:279–289, 1983.

[45] S. Homer and A. Selman. *Computability and Complexity Theory*. Springer Verlag, 2001.

[46] R. Impagliazzo. A personal view of average-case complexity theory. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 134–147. IEEE Computer Society Press, 1995.

[47] D. W. Juedes. *The complexity and distribution of computationally useful problems*. PhD thesis, Iowa State University, 1994.

[48] D. W. Juedes and J. Lutz. Weak completeness in E and E_2. *Theoretical Computer Science*, 143:149–158, 1995.

[49] C. Karg. LR(k) testing is average case complete. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 74–81, 1997.

[50] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[51] S. Kautz and P. Miltersen. Relative to a random oracle, NP is not small. *Journal of Computer and System Sciences*, 53(2):235–250, 1996.

[52] K. Ko. On the definition of some complexity classes of real numbers. *Mathematical Systems theory*, 16:95–109, 1983.

[53] K. Ko and D. Moore. Completeness, approximation and density. *SIAM Journal on Computing*, 10(4):787–796, Nov. 1981.

[54] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[55] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973. English translation of original in *Problemy Peredaci Informacii*.

[56] L. Levin. Average case complete problems. *SIAM Journal of Computing*, 15:285–286, 1986.

[57] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, 1997.

[58] L. Longpré and P. Young. Cook reducibility is faster than Karp reducibility. *Journal of Computer and System Sciences*, 41:389–401, 1990.

[59] J. Lutz. Category and measure in complexity classes. *SIAM Journal on Computing*, 19:1100–1131, 1990.

[60] J. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences*, 44:220–258, 1992.

[61] J. Lutz. The quantitative structure of exponential time. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 225–260. Springer-Verlag, New York, 1997.

[62] J. Lutz and E. Mayordomo. Cook versus Karp-Levin: Separating completeness notions if NP is not small. *Theoretical Computer Science*, 164:141–163, 1996.

[63] E. Mayordomo. Almost every set in exponential time is P-bi-immune. *Theoretical Computer Science*, 136:487–506, 1994.

[64] E. Mayordomo. *Contributions to the theory of resource-bounded measure*. PhD thesis, Universitat Politecnica de Catalunya, Barcelona, Spain, 1994.

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

[66] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[67] A. Pavan and A. Selman. Complete distributional problems, hard languages, and resource bounded measure. *Theoretical Computer Science*, 234:273–286, 2000.

[68] A. Pavan and A. Selman. Separation of NP-completeness notions. In *16th Annual IEEE Conference on Computational Complexity*, pages 78–89, 2001.

[69] A. Pavan and A. Selman. Separation of NP-completeness notions. *SIAM Journal on Computing*, To appear.

[70] M. Rabin. Degree of difficulty of computing a function and a partial ordering of recursive sets. Technical Report 2, The Hebrew University, Jerusalem, 1960.

[71] K. Regan and O. Watanabe. Personal communication.

[72] C. P. Schnorr. A unified approach to the definition of random sequences. *Mathematical Systems Theory*, 5:246–258, 1971.

[73] R. Schuler. Some properties of sets tractable under every polynomial-time computable distribution. *Information Processing Letters*, 55:179–184, 1995.

[74] R. Schuler. Truth-table closure and Turing closure of average polynomial time have different measures in EXP. In *Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity*, pages 190–197, 1996.

[75] R. Schuler and T. Yamakami. Sets computable in polynomial time on the average. In *Proceedings of the First Annual International Computing and Combinatorics Conference, Lecture Notes in Computer Science*, volume 959, pages 650–661. Springer-Verlag, 1995.

[76] R. Schuler and T. Yamakami. Structural average case complexity. *Journal of Computer and System Sciences*, 52:308–327, 1996.

[77] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.

[78] A. Selman. Reductions on NP and P-selective sets. *Theoretical Computer Science*, 19:287–304, 1982.

[79] A. Selman. A survey of one-way functions in complexity theory. *Mathematical Systems Theory*, 25:203–221, 1992.

[80] I. Simon and J. Gill. Polynomial reducibilities and upward diagonalizations. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*, pages 186–194, 1977.

[81] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[82] S. Toda. On polynomial-time truth-table reducibilities of intractable sets to P-selective sets. *Mathematical Systems Theory*, 24(2):69–82, 1991.

[83] A. Turing. On computable numbers with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–365, 1936.

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

[85] R. Venkatesan and L. Levin. Random instances of a graph coloring problem are hard. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 217–222, 1988.

[86] R. Venkatesan and S. Rajagopalan. Average case intractability of diophantine and matrix problems. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, pages 632–642, 1992.

[87] J. Wang. Average-case completeness of a word problem for groups. In *Proceedings of the Twenty-Seventh ACM Symposium on Theory of Computing*, pages 325–334, 1995.

[88] J. Wang. Average-case computational complexity theory. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 295–328. Springer-Verlag, 1997.

[89] J. Wang and J. Belanger. On the NP-isomorphism problem with respect to random instances. *Journal of Computer and System Sciences*, 50:151–164, 1995.

[90] O. Watanabe. A comparison of polynomial time completeness notions. *Theoretical Computer Science*, 54:249–265, 1987.

[91] H. Wilf. An O(1) expected time algorithm for graph coloring problem. *Information Processing Letters*, 18:119–122, 1984.

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