The complexity of parameters for probabilistic and quantum computation

by

Josef Michael Breutzmann

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Computer Science

Major Professor: Jack H. Lutz

Iowa State University
Ames, Iowa

1996

Graduate College
Iowa State University

This is to certify that the Doctoral dissertation of

Josef Michael Breutzmann

has met the dissertation requirements of Iowa State University

Committee Member

Committee Member

Committee Member

Committee Member

Major Professor

For the Major Program

For the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to acknowledge the assistance and support of my colleagues in the Mathematics, Computer Science and Physics Department at Wartburg College and the members of my program of study committee.

I would also like to acknowledge the support, assistance and sacrifices of the members of my family. The many hours spent apart from them would have been intolerable without their active encouragement and support for this undertaking. In particular, the support of my wife, Julie, may have gone unacknowledged at times but never unnoticed nor unappreciated.

Finally I would like to thank my advisor, Jack H. Lutz. Without his input, support and patience this dissertation would never have come to fruition. He has been very supportive of my activities and extremely understanding of the trials of being a non-traditional graduate student. I acknowledge a very significant debt of gratitude to him personally and professionally.

ABSTRACT

In this thesis we study some effects of allowing computational models that use parameters whose own computational complexity has a strong effect on the computational complexity of the languages computable from the model. We show that in the probabilistic and quantum models there are parameter sets that allow one to obtain noncomputable outcomes.

In Chapter 3 we define $\mathrm{BP}_\beta\mathrm{P}$ the BPP class based on a coin with bias β. We then show that if β is BPP-computable then it is the case that $\mathrm{BP}_\beta\mathrm{P} = \mathrm{BPP}$. We also show that each language L in P/CLog is in $\mathrm{BP}_\beta\mathrm{P}$ for some β. Hence there are some β from which we can compute noncomputable languages. We also examine the robustness of the class BPP with respect to small variations from fairness in the coin.

In chapter 4 we consider measures that are based on polynomial-time computable sequences of biased coins in which the biases are bounded away from both zero and one (strongly positive P-sequences). We show that such a sequence $\vec{\beta}$ generates a measure $\mu^{\vec{\beta}}$ equivalent to the uniform measure in the sense that if $\mathcal{C}$ is a class of languages closed under positive, polynomial-time, truth-table reductions with queries of linear length then $\mathcal{C}$ has $\mu^{\vec{\beta}}$-measure zero if and only if it has measure zero relative to the uniform measure μ. The classes P, NP, BPP, P/Poly, PH, and PSPACE are among those to which this result applies. Thus the measures of these much-studied classes are robust with respect to changes of this type in the underlying probability measure.

In chapter 5 we introduce the quantum computation model and the quantum complexity class BQP. We claim that the computational complexity of the amplitudes is

a critical factor in determining the languages computable using the quantum model. Using results from chapter 3 we show that the quantum model can also compute non-computable languages from some amplitude sets. Finally, we determine a restriction on the amplitude set to limit the model to the range of languages implicit in others' typical meaning of the class BQP.

CHAPTER 1 INTRODUCTION

This thesis deals with aspects of the use of randomness in computation. In particular Chapter 3 deals with the use of biased coins in computation. It explores the possibility that the value of the bias might contain encoded information that can be extracted through repeated tosses of the coin. In chapter 4 we explore probability measures derived from an infinite sequence of biased coins each of which has a potentially different bias. We examine a condition under which two different measures derived from two different such sequences agree on a certain collection of measure zero sets. In Chapter 5 we relate the results in Chapter 3 to the quantum computation model. There again we show that the amplitudes, the values that correspond to the probabilities in the probabilistic model, can contain encoded information that can be extracted during computation.

Chapters 3, 4, and 5 are introduced separately in the following three sections of this introduction.

1.1 Probabilistic Computation Using a Biased Coin

We consider the languages in the class BPP to represent what may be feasibly computed. These algorithms require time polynomial in the input size and a source of independent, uniformly random bits. Both requirements seem reasonable and practical and hence our identification of BPP languages as being feasible. The first requirement is the traditional definition of feasible computation. The second allows the algorithm to use randomness to compute an answer that is correct with high probability. This answer has only an exponentially small probability of being incorrect and hence is quite adequate for many purposes.

Working on factoring and primality testing, Berlekamp [11], Solovay and Strassen [72], and Rabin [56] were among the first to give probabilistic algorithms. Attacking the subject from the direction of probabilistic Turing machines, de Leeuw *et. al.* [24], Santos [60], and Gill [30, 31] were the pioneers. In addition, Gill was the first to define the class BPP. The BP operator was defined by Schöning [65] to study not only the class BPP but also AM, the Arthur-Merlin class of Babai [4].

In Chapter 3 of this thesis we shall recall and use Schöning's operator/witness version of BPP. To motivate the quantum model in Chapter 6 we shall introduce an equivalent time-evolution operator formulation that uses a matrix multiplication technique to model a probabilistic Turing machine. It has more flexibility and provides a good intuitive introduction to the notation and ideas used with quantum computation.

The standard definition of BPP relies on the uniform probability measure in one way or another depending on the formalization being used. This uniformity is introduced through fair coins in the Turing machine model, through probability measures in the operator model and through fair martingales in measure theoretic approaches. In Chapter 3 we will modify the definition of the class BPP to use non-uniform distributions. In particular we will consider classes analogous to BPP that are defined using measures based on independent tosses of a coin with bias β for some real number beta. If β is equal to $\frac{1}{2}$, the result is BPP.

The complexity of the real parameter β will be shown to influence the complexity of the corresponding probabilistic class $\mathrm{BP}_\beta\mathrm{P}$. We will show that $\mathrm{BPP} = \mathrm{BP}_\beta\mathrm{P}$ in the case that β polynomial-time computable. That is, if β is not too complex then a fair coin can simulate a coin with bias β. We will also define a new complexity class for real numbers, $\mathrm{BPP}_{\mathrm{CF}}$. This class consists of real numbers computable by a BPP computation. For this class of β's we will also show that $\mathrm{BPP} = \mathrm{BP}_\beta\mathrm{P}$.

The intuition that led us into this investigation was that one ought to be able to compute more complex languages given a more complex bias. We show in Chapter 3 that in fact there are biases that allow us to compute non-recursive languages. The proof of this fact is achieved using advice classes. We show that each language in the advice class P/CLog is contained in some $\mathrm{BP}_\beta\mathrm{P}$ class and that each $\mathrm{BP}_\beta\mathrm{P}$ class in turn is contained in P/Poly. Since P/CLog contains non-recursive languages we will have shown that for some β at least $\mathrm{BPP} \neq \mathrm{BP}_\beta\mathrm{P}$.

In the final section of Chapter 3 we shall note that $\mathrm{BP}_\beta\mathrm{P}$ algorithms can be very sensitive to small variations in the distribution of the random bits. Since any real-world source of random bits, biased or unbiased, is unlikely to provide perfect distributions, we will propose a definition of robust-$\mathrm{BP}_\beta\mathrm{P}$, a robust form of $\mathrm{BP}_\beta\mathrm{P}$ which is insensitive to small variations in the distribution. Many investigators [23, 59, 76, 75, 81, 82] have shown that various similar robust forms of BPP are equivalent to the standard BPP. That is, every language in BPP has a robust algorithm which can be used to decide it. We will show that robust-$\mathrm{BP}_\beta\mathrm{P}$ is also equal to BPP. This strengthens our intuition that BPP represents the class of feasibly computable languages.

1.2 Complexity Classes Under Measures Based on Sequences of Biased Coins

Chapter 4 of this thesis is joint work with Lutz[20].

Intuitively, suppose that a language $A \subseteq \{0,1\}^*$ is chosen according to a random experiment in which an independent toss of a fair coin is used to decide whether each string is in A. Then *classical* Lebesgue measure theory (described in [33, 54], for example) identifies certain *measure 0* sets X of languages, for which the probability that $A \in X$ in this experiment is 0. *Effective* measure theory, which says what it means for a set of decidable languages to have measure 0 as a subset of the set of all such languages, has been investigated by Freidzon [29], Mehlhorn [52], and others. The *resource-bounded* measure theory introduced by Lutz [43] is a powerful generalization of Lebesgue measure. Special cases of resource-bounded measure include classical Lebesgue measure; a strengthened version of effective measure; and most importantly, measures in $\mathrm{E} = \mathrm{DTIME}(2^{\mathrm{linear}})$, $\mathrm{E}_2 = \mathrm{DTIME}(2^{\mathrm{polynomial}})$, and other complexity classes. The *small* subsets of such a complexity class are then the measure 0 sets; the *large* subsets are the measure 1 sets (complements of measure 0 sets). We say that *almost every* language in a complexity class $\mathcal{C}$ has a given property if the set of languages in $\mathcal{C}$ that exhibit the property has measure 1 in $\mathcal{C}$.

All work to date on the measure-theoretic structure of complexity classes has employed the resource-bounded measure that is described briefly and intuitively above. This resource-bounded measure is based on the *uniform* probability measure, corresponding to the fact that the coin tosses are fair and independent in the above-described random experiment. The uniform probability measure has been a natural and fruitful starting point for the investigation of resource-bounded measure (just as it was for the investigation of classical measure), but there are good reasons to also investigate resource bounded measures that are based on other probability measures. For example, the study of such alternative resource-bounded measures may be expected to have the following benefits.

(i) The study will enable us to determine which results of resource-bounded measure are particular to the uniform probability measure and which are not. This, in turn, will provide some criteria for identifying contexts in which the uniform probability measure is, or is not, the natural choice.

(ii) The study is likely to help us understand how the complexity of the underlying probability measure interacts with other complexity parameters, especially in such areas as algorithmic information theory, average case complexity, cryptography, and computational learning, where the variety of probability measures already plays a major role.

(iii) The study will provide new tools for proving results concerning resource-bounded measure based on the uniform probability measure.

Chapter Four initiates the study of resource-bounded measures that are based on nonuniform probability measures.

Let $\{0,1\}^\infty$ be the set of all languages $A \subseteq \{0,1\}^*$. (The set $\{0,1\}^\infty$ is often called *Cantor space.*) Given a probability measure ν on $\{0,1\}^\infty$ (a term defined precisely below), section 4.1 of this paper describes the basic ideas of resource-bounded ν-measure, generalizing definitions and results from [43, 45, 44] to ν in a natural way. In particular, section 4.1 specifies what it means for a set $X \subseteq \{0,1\}^\infty$ to have p-ν-measure 0 (written $\nu_{\mathrm{p}}(X) = 0$), p-ν-measure 1, ν-measure 0 in E (written $\nu(X|\mathrm{E}) = 0$), ν-measure 1 in E, ν-measure 0 in E_2, or ν-measure 1 in E_2.

Most of the results in Chapter Four concern a restricted (but broad) class of probability measures on $\{0,1\}^\infty$, namely, coin-toss probability measures that are given by P- computable, strongly positive sequences of biases. These probability measures are described intuitively in the following paragraphs (and precisely in Chapter 2).

Given a sequence $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \ldots)$ of real numbers (biases) $\beta_i \in [0,1]$, the *coin-toss probability measure* (also call the *product probability measure*) *given by* $\vec{\beta}$ is the probability measure $\mu^{\vec{\beta}}$ on $\{0,1\}^\infty$ that corresponds to the random experiment in which a language $A \in \{0,1\}^\infty$ is chosen probabilistically as follows. For each string s_i in the standard enumeration $s_0, s_1, s_2, \ldots$ of $\{0,1\}^*$, we toss a special coin, whose probability is β_i of coming up heads, in which case $s_i \in A$, and $1 - \beta_i$ of coming up tails, in which case $s_i \notin A$. The coin tosses are independent of one another.

In the special case where $\vec{\beta} = (\beta, \beta, \beta, \ldots)$, i.e., the biases in the sequence $\vec{\beta}$ are all β, we write μ^β for $\mu^{\vec{\beta}}$. In particular, $\mu^{\frac{1}{2}}$ is the uniform probability measure, which, in the literature of resource-bounded measure, is denoted simply by μ.

A sequence $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \ldots)$ of biases is *strongly positive* if there is a real number $\delta > 0$ such that each $\beta_i \in [\delta, 1-\delta]$. The sequence $\vec{\beta}$ is P-*computable* (and we call it a P-*sequences of biases*) if there is a polynomial-time algorithm that, on input $(s_i, 0^r)$, computes a rational approximation of β_i to within 2^{-r}.

In section 4.2, we prove the Summable Equivalence Theorem, which implies that, if $\vec{\alpha}$ and $\vec{\beta}$ are strongly positive P-sequences of biases that are "close" to one another, in the sense that $\sum_{i=0}^{\infty} |\alpha_i - \beta_i| < \infty$, then for every set $X \subseteq \{0,1\}^{\infty}$,

$$\mu_{\mathrm{p}}^{\vec{\alpha}}(X) = 0 \Longleftrightarrow \mu_{\mathrm{p}}^{\vec{\beta}}(X) = 0.$$

That is, the p-measure based on $\vec{\alpha}$ and the p- measure based on $\vec{\beta}$ are in absolute agreement as to which sets of languages are small.

In general, if $\vec{\alpha}$ and $\vec{\beta}$ are not in some sense close to one another, then the p-measures based on $\vec{\alpha}$ and $\vec{\beta}$ need not agree in the above manner. For example, if $\alpha, \beta \in [0,1]$, $\alpha \neq \beta$, and

$$X_\alpha = \left\{ A \in \{0,1\}^{\infty} | \lim_{n\to\infty} 2^{-n} |A \cap \{0,1\}^n| = \alpha \right\},$$

then a routine extension of the Weak Stochasticity Theorem of [47] shows that $\mu_{\mathrm{p}}^{\alpha}(X_\alpha) = 1$, while $\mu_{\mathrm{p}}^{\beta}(X_\alpha) = 0$.

Notwithstanding this example, many applications of resource- bounded measure do not involve *arbitrary* sets $X \subseteq \{0,1\}^{\infty}$, but rather are concerned with the measures of *complexity classes* and other closely related classes of languages. Many such classes of interest, including P, NP, co-NP, R, BPP, AM, P/Poly, PH, PSPACE, etc., are closed under positive, polynomial-time truth-table reductions ($\leq_{\mathrm{pos-tt}}^{\mathrm{P}}$-reductions), and their intersections with E are closed under $\leq_{\mathrm{pos-tt}}^{\mathrm{P}}$-reductions with linear bounds on the lengths of the queries ($\leq_{\mathrm{pos-tt}}^{\mathrm{P,lin}}$-reductions).

The main theorem of Chapter 4 is the Bias Equivalence Theorem. This result, proven in section 4.6, says that, for every class $\mathcal{C}$ of languages that is closed under

$\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions, the p-measure of $\mathcal{C}$ is somewhat robust with respect to changes in the underlying probability measure. Specifically, if $\vec{\alpha}$ and $\vec{\beta}$ are strongly positive P-sequences of biases and $\mathcal{C}$ is a class of languages that is closed under $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions, then the Bias Equivalence Theorem says that

$$\mu^{\vec{\alpha}}_{\mathrm{p}}(\mathcal{C}) = 0 \iff \mu^{\vec{\beta}}_{\mathrm{p}}(\mathcal{C}) = 0.$$

To put the matter differently, for every strongly positive P-sequence $\vec{\beta}$ of biases and every class $\mathcal{C}$ that is closed under $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions,

$$\mu^{\vec{\beta}}_{\mathrm{p}}(\mathcal{C}) = 0 \iff \mu_{\mathrm{p}}(\mathcal{C}) = 0.$$

This result implies that most applications of resource-bounded measure to date can be immediately generalized from the uniform probability measure (in which they were developed) to arbitrary coin-toss probability measures given by strongly positive P-sequences of biases.

The Bias Equivalence Theorem also offers the following new technique for proving resource-bounded measure results. If $\mathcal{C}$ is a class that is closed under $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions, then in order to prove that $\mu_{\mathrm{p}}(\mathcal{C}) = 0$, it suffices to prove that $\mu^{\vec{\beta}}_{\mathrm{p}}(\mathcal{C}) = 0$ for some conveniently chosen strongly positive P-sequence $\vec{\beta}$ of biases. (The Bias Equivalence Theorem has already been put to this use in the forthcoming paper [48].)

The plausibility and consequences of the hypothesis $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ are subjects of recent and ongoing research [51, 47, 36, 49, 46, 21, 48]. The Bias Equivalence Theorem immediately implies that the following three statements are equivalent.

(H1) $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$.

(H2) For every strongly positive P-sequence $\vec{\beta}$ of biases, $\mu^{\vec{\beta}}_{\mathrm{p}}(\mathrm{NP}) \neq 0$.

(H3) There exists a strongly positive P-sequence $\vec{\beta}$ of biases such that $\mu^{\vec{\beta}}_{\mathrm{p}}(\mathrm{NP}) \neq 0$.

The statements (H2) and (H3) are thus new, equivalent formulations of the hypothesis (H1).

The proof of the Bias Equivalence Theorem uses three main tools. The first is the Summable Equivalence Theorem, which we have already discussed. The second is the Martingale Dilation Theorem, which is proven in section 4.4. This result concerns martingales (defined in section 4.1), which are the betting algorithms on which resource-bounded measure is based. Roughly speaking, the Martingale Dilation Theorem gives a method of transforming ("dilating") a martingale for one coin-toss probability measure into a martingale for another, perhaps very different, coin-toss probability measure, provided that the former measure is obtained from the latter via an "orderly" truth-table reduction.

The third tool is the Positive Bias Reduction Theorem, which is presented in section 4.5. If $\vec{\alpha}$ and $\vec{\beta}$ are two strongly positive sequences of biases that are exactly P-computable (with no approximation), then the *positive bias reduction* of $\vec{\alpha}$ to $\vec{\beta}$ is a truth-table reduction (in fact, an orderly $\leq_{\mathrm{pos-tt}}^{\mathrm{P,lin}}$-reduction) that uses the sequence $\vec{\beta}$ to "approximately simulate" the sequence $\vec{\alpha}$. It is especially crucial for our main result that this reduction is efficient and positive. (The circuits constructed by the truth-table reduction contain AND gates and OR gates, but no NOT gates.)

The Summable Equivalence Theorem, the Martingale Dilation Theorem, and the Positive Bias Reduction Theorem are only developed and used here as tools to prove our main result. Nevertheless, these three results are of independent interest, and are likely to be useful in future investigations.

1.3 Parameter Complexity for Quantum Computation

Quantum computation is a topic receiving a lot of attention in recent years. If its techniques can be implemented in physical machines it will provide an important new computational paradigm for our use.

The earlier work on quantum models[28, 25, 12] received a boost recently when Shor [67] gave polynomial-time, quantum algorithms for two very well-known and very important problems, factoring and the discrete logarithm. These problems were previously suspected to be not polynomial-time computable with deterministic techniques.

Following others in the area [12], in Chapter 5 we will provide an introduction to quantum computation based on probabilistic computation. Whereas each transition in a probabilistic computation has a probability, *i.e.*, a real number between 0 and 1, associated with it, in a quantum computation we associate with each transition a complex number called the amplitude. These amplitudes play a role very similar to the biases for the coins we have used to define probability distributions in Chapters 3 and 4.

Paralleling what we did in Chapter 3, we will show that one can encode information into the amplitudes that can later be extracted during the computation. This allows one to compute noncomputable languages using quantum methods.

We believe that the basic definitions of the field need to be revised to take into account this phenomenon. The complexity of the amplitudes needs to be one of the basic parameters of the classes defined by quantum methods. Other workers in the field are slowly coming to this realization. Recent papers ([9] and [67] for example) suggest that the amplitudes satisfy a computability requirement similar to the one we propose. That is, the basic class on quantum computation, BQP, should be defined in terms of amplitudes for which on can compute logarithmically many bits in polynomial time.

We extend this idea by defining classes BQP[S] for sets of amplitudes S other than that proposed above. In Chapter 5 we also provide a link between BP$_\beta$P and BQP by showing that BP$_\beta$P $\subseteq$ BQP[$\{0, \beta, 1-\beta, 1\}$]. This allows us to carry our results from probabilistic computation over to quantum computation. In particular, that BQP[S] contains noncomputable languages for some choices of S.

Whereas in Chapter 3 we will show that P/CLog and P/Poly provide lower and upper bounds for $\mathrm{BP}_\beta\mathrm{P}$, in Chapter 5 we show that $\mathrm{BQP}[\mathrm{E}_{\mathrm{CF}}] \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$. Since our class $\mathrm{BQP}[\mathrm{E}_{\mathrm{CF}}]$ corresponds to the 'usual' quantum polynomial-time language class BQP, we will have shown that the languages of BQP are computable.

CHAPTER 2 PRELIMINARIES AND NOTATION

2.1 Basic Definitions and Notation

In this thesis, $\mathbf{N}$ denotes the set of all nonnegative integers, $\mathbf{Z}$ denotes the set of all integers, $\mathbf{Z}^+$ denotes the set of all positive integers, $\mathbf{Q}$ denotes the set of all rational numbers, $\mathbf{D}$ denotes the set of all dyadic rational numbers(*i.e.* those expressible with a power of two in the denominator), $\mathbf{R}$ denotes the set of all real numbers and $\mathbf{C}$ denotes the set of all complex numbers.

We are interested in the computability of real and complex numbers in the sense given by Ko [41]. We say that a function $\varphi : \{0\}^* \to \mathbf{D}$ is a *Cauchy function* for a real number x if for every $n \in \mathbf{N}$ we have $|\varphi(0^n) - x| \leq 2^{-n}$. For a given real number x we call the set of all such functions $CF(x)$. If $CF(x)$ contains a polynomial-time computable function then we say x is polynomial-time computable. We denote the set of all polynomial-time computable real numbers by $\mathrm{P_{CF}}$. We define the class of exponential-time computable real numbers $\mathrm{E_{CF}}$ as the set of real numbers x for which there is a function in $CF(x)$ which is computable in $O(2^{\text{linear}})$ time. A complex number is polynomial-time (exponential-time) computable if both its real and its imaginary parts are polynomial-time (exponential-time) computable.

We write $\{0,1\}^*$ for the set of all (finite, binary) *strings*, and we write $|x|$ for the length of a string x. The empty string, λ, is the unique string of length 0. The *standard enumeration* of $\{0,1\}^*$ is the sequence $s_0 = \lambda, s_1 = 0, s_2 = 1, s_3 = 00, \ldots$, ordered first by length and then lexicographically. For $x, y \in \{0,1\}^*$, we write $x < y$ if x precedes y

in this standard enumeration. For $n \in \mathbf{N}$, $\{0,1\}^n$ denotes the set of all strings of length n, and $\{0,1\}^{\leq n}$ denotes the set of all strings of length at most n.

If x is a string or an (infinite, binary) *sequence*, and if $0 \leq i \leq j < |x|$, then $x[i..j]$ is the string consisting of the i^{th} through j^{th} bits of x. In particular, $x[0..i-1]$ is the *i-bit prefix* of x. We write $x[i]$ for $x[i..i]$, the i^{th} bit of x. (Note that the leftmost bit of x is $x[0]$, the 0^{th} bit of x.)

If w is a string and x is a string or sequence, then we write $w \sqsubseteq x$ if w is a prefix of x, i.e., if there is a string or sequence y such that $x = wy$.

The *Boolean value* of a condition ϕ is $[\![\phi]\!] = \textbf{if } \phi \textbf{ then } 1 \textbf{ else } 0$.

In this thesis we use both the binary logarithm $\log \alpha = \log_2 \alpha$ and the natural logarithm $\ln \alpha = \log_e \alpha$.

Many of the functions in this thesis are real-valued functions on discrete domains. These typically have the form

$$f : \mathbf{N}^d \times \{0,1\}^* \longrightarrow \mathbf{R}, \tag{2.1}$$

where $d \in \mathbf{N}$. (If $d = 0$, we interpret this to mean that $f : \{0,1\}^* \longrightarrow \mathbf{R}$.) Such a function f is defined to be p-*computable* if there is a function

$$\hat{f} : \mathbf{N} \times \mathbf{N}^d \times \{0,1\}^* \longrightarrow \mathbf{Q} \tag{2.2}$$

with the following two properties.

(i) For all $r, k_1, \ldots, k_d \in \mathbf{N}$ and $w \in \{0,1\}^*$,

$$|\hat{f}(r, k_1, \ldots, k_d, w) - f(k_1, \ldots, k_d, w)| \leq 2^{-r}.$$

(ii) There is an algorithm that, on input $(r, k_1, \ldots, k_d, w)$, computes the value $\hat{f}(r, k_1, \ldots, k_d, w)$ in $(r + k_1 + \ldots + k_d + |w|)^{O(1)}$ time.

Similarly, f is defined to be p_2-*computable* if there is a function $\hat{f}$ as in (2.2) that satisfies condition (i) above and the following condition.

(ii′) There is an algorithm that, on input $(r, k_1, \ldots, k_d, w)$, computes the value $\hat{f}(r, k_1, \ldots, k_d, w)$ in $2^{\log(r+k_1+\ldots+k_d+|w|)^{O(1)}}$ time.

In this thesis, functions of the form (2.1) frequently have the form

$$f : \mathbf{N}^d \times \{0,1\}^* \longrightarrow [0, \infty)$$

or the form

$$f : \mathbf{N}^d \times \{0,1\}^* \longrightarrow [0, 1].$$

If such a function is p-computable or p_2-computable, then we assume without loss of generality that the approximating function $\hat{f}$ of (2.2 actually has the form

$$\hat{f} : \mathbf{N} \times \mathbf{N}^d \times \{0,1\}^* \longrightarrow \mathbf{Q} \cap [0, \infty)$$

or the form

$$\hat{f} : \mathbf{N} \times \mathbf{N}^d \times \{0,1\}^* \longrightarrow \mathbf{Q} \cap [0, 1],$$

respectively.

2.2 Probability Measures on Cantor Space

In this section, we develop basic elements of resource-bounded measure based on an arbitrary (Borel) probability measure ν. The ideas here generalize the corresponding ideas of "ordinary" resource-bounded measure (based on the uniform probability measure μ) in a straightforward and natural way, so our presentation is relatively brief. The reader is referred to [43, 44] for additional discussion.

We work in the *Cantor space* $\{0,1\}^\infty$, consisting of all languages $A \subseteq \{0,1\}^*$. We identify each language A with its *characteristic sequence*, which is the infinite binary sequence χ_A defined by

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

for each $n \in \mathbf{N}$. Relying on this identification, we also consider $\{0,1\}^\infty$to be the set of all infinite binary sequences.

For each string $w \in \{0,1\}^*$, the *cylinder generated by* w is the set

$$\mathbf{C}_w = \{A \in \{0,1\}^\infty \mid w \sqsubseteq \chi_A\}.$$

Note that $\mathbf{C}_\lambda = \{0,1\}^\infty$.

We first review the well-known notion of a (Borel) probability measure on $\{0,1\}^\infty$.

Definition. A *probability measure* on $\{0,1\}^\infty$is a function

$$\nu : \{0,1\}^* \longrightarrow [0,1]$$

such that $\nu(\lambda) = 1$, and for all $w \in \{0,1\}^*$,

$$\nu(w) = \nu(w0) + \nu(w1).$$

Intuitively, $\nu(w)$ is the probability that $A \in \mathbf{C}_w$ when we "choose a language $A \in \{0,1\}^\infty$ according to the probability measure ν." We sometimes write $\nu(\mathbf{C}_w)$ for $\nu(w)$.

It is easy to see that for each $n \in \mathbf{N}$, this definition also gives a probability measures on $\{0,1\}^n$, *i.e.*, $\sum_{|w|=n} \nu(w) = 1$. We shall use the symbol $\nu(w)$ to represent both the probability measure on $\{0,1\}^\infty$ and for each n, the probability measure on $\{0,1\}^n$ where $|w| = n$. The usage will be clear from context.

Examples.

1. The *uniform probability measure* μ is defined by

$$\mu(w) = 2^{-|w|}$$

for all $w \in \{0,1\}^*$.

2. A *sequence of biases* is a sequence $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \ldots)$, where each $\beta_i \in [0,1]$. Given a sequence of biases $\vec{\beta}$, the $\vec{\beta}$*coin-toss probability measure* (also called the

$\vec{\beta}$*-product probability measure*) is the probability measure $\mu^{\vec{\beta}}$ defined by

$$\mu^{\vec{\beta}}(w) = \prod_{i=0}^{|w|-1} ((1-\beta_i)\cdot(1-w[i]) + \beta_i \cdot w[i])$$

for all $w \in \{0,1\}^*$.

3. If $\beta = \beta_0 = \beta_1 = \beta_2 = \ldots$, then we write μ^β for $\mu^{\vec{\beta}}$. In this case, we have the simpler formula

$$\mu^\beta(w) = (1-\beta)^{\#(0,w)} \cdot \beta^{\#(1,w)},$$

where $\#(b,w)$ denotes the number of b's in w. Note that $\mu^{\frac{1}{2}} = \mu$.

Intuitively, $\mu^{\vec{\beta}}(w)$ is the probability that $w \sqsubseteq A$ when the language $A \subseteq \{0,1\}^*$ is chosen probabilistically according to the following random experiment. For each string s_i in the standard enumeration $s_0, s_1, s_2, \ldots$ of $\{0,1\}^*$, we (independently of all other strings) toss a special coin, whose probability is β_i of coming up heads, in which case $s_i \in A$, and $1-\beta_i$ of coming up tails, in which case $s_i \notin A$.

<u>Definition</u>. A probability measure ν on $\{0,1\}^\infty$is *positive* if, for all $w \in \{0,1\}^*$, $\nu(w) > 0$.

<u>Definition</u>. If ν is a positive probability measure and $u, v \in \{0,1\}^*$, then the *conditional* ν*-measure of* u *given* v is

$$\nu(u|v) = \begin{cases} 1 & \text{if } u \sqsubseteq v \\ \frac{\nu(u)}{\nu(v)} & \text{if } v \sqsubseteq u \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\nu(u|v)$ is the conditional probability that $A \in \mathbf{C}_u$, given that $A \in \mathbf{C}_v$, when $A \in \{0,1\}^\infty$ is chosen according to the probability measure ν.

Most of this thesis concerns the following special type of probability measure.

<u>Definition</u>. A probability measure ν on $\{0,1\}^\infty$is *strongly positive* if (ν is positive and) there is a constant $\delta > 0$ such that, for all $w \in \{0,1\}^*$ and $b \in \{0,1\}$, $\nu(wb|w) \geq \delta$.

<u>Definition</u>. A sequence of biases $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \ldots)$ is *strongly positive* if there is a constant $\delta > 0$ such that, for all $i \in \mathbf{N}$, $\beta_i \in [\delta, 1-\delta]$.

If $\vec{\beta}$ is a sequence of biases, then the following two observations are clear.

1. $\mu^{\vec{\beta}}$ is positive if and only if $\beta_i \in (0,1)$ for all $i \in \mathbf{N}$.

2. If $\mu^{\vec{\beta}}$ is positive, then for each $w \in \{0,1\}^*$,

$$\mu^{\vec{\beta}}(w0|w) = 1 - \beta_{|w|}$$

and

$$\mu^{\vec{\beta}}(w1|w) = \beta_{|w|}.$$

It follows immediately from these two things that the probability measure $\mu^{\vec{\beta}}$ is strongly positive if and only if the sequence of biases $\vec{\beta}$ is strongly positive.

In this thesis, we are primarily interested in strongly positive probability measures ν that are p-computable in the sense above.

CHAPTER 3 PROBABILISTIC COMPUTATION UNDER BIASED COIN DISTRIBUTIONS

The following is a standard definition of the class BPP of bounded-error, probabilistic, polynomial-time computable languages.

Definition.

A language $L \subseteq \{0,1\}^*$ is in the class BPP if and only if there exist a language $A \in \mathrm{P}$ and a polynomial $q(n)$ such that for every $x \in \{0,1\}^*$,

$$\frac{|\{w \in \{0,1\}^{q(|x|)} | < x, w > \in A \iff x \in L\}|}{2^{q(|x|)}} > \frac{3}{4}.$$

In this definition the strings designated by w, called the *witnesses* for or against x, can be thought of as sequences of random bits or coin tosses. Intuitively, the definition says that of all possible witnesses, at least three-quarters of them give the right answer when 'asked' whether x is in L. The equation can be restated in terms of the uniform distribution as $\mu\left(\{w \in \{0,1\}^{q(|x|)} | < x, w > \in A \iff x \in L\}\right) > \frac{3}{4}$.

There is nothing unique about the fraction $\frac{3}{4}$ in the definition. It could just as easily be any real constant strictly greater than $\frac{1}{2}$ and less than 1. The $\frac{3}{4}$ can also be amplified to $1 - 2^{-p(|x|)}$ for an arbitrary polynomial [66].

The class BPP is frequently defined in terms of coin-tossing Turing machines. These represent polynomial-time algorithms that may depend on the result of the flip of a fair coin to determine their next action. All standard definitions of BPP depend on the fact that the coin is fair or the witness string is chosen uniformly at random from the set of all strings of the given length.

In this chapter we investigate the results of using a biased coin, *i.e.*, one with probability β of producing a one and complementary probability $1-\beta$ of producing zero. We will see that while a biased coin can always compute those languages computable with a fair coin, the reverse is not true. We will show that some biased coins can compute languages that a fair coin cannot.

In the last section of this chapter we investigate variable-bias coins. These are sources of random bits whose probability of producing a one are not known precisely or that vary within certain limits. We will show that there is a robust definition of the class BPP that is resistant to such less-than-perfect random sources.

3.1 The Class $\mathrm{BP}_\beta\mathrm{P}$

We are interested in BPP classes that are based on probability distributions other than the uniform distribution. We can define these measure-dependent versions of BPP as follows.

Definition.

Let ν be a probability measure on $\{0,1\}^\infty$. A language $L \subseteq \{0,1\}^*$ is in the class $\mathrm{BP}_\nu\mathrm{P}$ if and only if there exist a language $A \in \mathrm{P}$ and a polynomial $q(n)$ such that for every $x \in \{0,1\}^*$,

$$\nu(\{w \in \{0,1\}^{q(|x|)} | < x, w >\in A \iff x \in L\}) > \frac{3}{4}.$$

Again, this says that three-quarters of the witnesses give the correct answer when asked if x is in L. However this time the measure determining what constitutes three-fourths is given by ν.

In Chapter 2 we defined β-biased-coin probability measures, μ^β. They independently give each bit of a witness string probability β of being 1 and probability $1-\beta$ of being 0 for some $0 < \beta < 1$.

We will write $\mathrm{BP}_\beta\mathrm{P}$ for $\mathrm{BP}_{\mu^\beta}\mathrm{P}$ to designate the bounded error, probabilistic, polynomial time class given by the β-biased-coin probability measure. It is clearly the case that $\mathrm{BPP} = \mathrm{BP}_{\frac{1}{2}}\mathrm{P}$. In the remainder of this section we will investigate the relationship between $\mathrm{BP}_\beta\mathrm{P}$ and BPP. We shall do this in terms of the resource-bounded computability of the bias β.

As is typical for probabilistic classes, it will be convenient to have an amplified version of the definition. This is one in which the constant $\frac{3}{4}$ is replaced by $1 - 2^{-p(|x|)}$ for an arbitrary polynomial p. This is achieved by running polynomially many simulations of the computation and taking the majority answer.

In proving the amplification result just mentioned, as well as in proving many of the other lemmas and theorems in this section, we will find the Chernoff bounds useful. They put limits on the probability that the actual number of ones in a μ^β-random string is very different from the expected number of ones. If a string w is of length n and β is the probability that any given bit is 1, then the expected number of 1's in w is $n\beta$.

Theorem 3.1.1 (Chernoff). [22, 32]

If S is the number of 1's in a string of length n and β is the probability that any one of the bits is a 1, then

$$Pr\,(S \leq (1-\varepsilon)n\beta) \leq e^{-\varepsilon^2 n\beta/2}, \text{ and}$$

$$Pr\,(S \geq (1+\varepsilon)n\beta) \leq e^{-\varepsilon^2 n\beta/3}.$$

Our first use of the Chernoff bounds will be to prove the following $\mathrm{BP}_\beta\mathrm{P}$ amplification lemma.

Lemma 3.1.2 ($\mathrm{BP}_\beta\mathrm{P}$ Amplification).

If $L \in \mathrm{BP}_\beta\mathrm{P}$ and r is any polynomial, then there exist a language $A \in P$ and a polynomial q such that for every $x \in \{0,1\}^*$ we have

$$\mu^\beta(\{w \in \{0,1\}^{q(|x|)} \mid \, < x, w > \in A \iff x \in L\}) \geq 1 - 2^{-r(|x|)}.$$

Proof.

Let a polynomial $r(n)$ be given. Let $t(n) = 48r(n)$ and let B and $s(n)$ witness that $L \in \mathrm{BP}_\beta\mathrm{P}$. *i.e.*, $\mu^\beta(\{w \in \{0,1\}^{s(|x|)} | < x, w > \in B \iff x \in L\}) > \frac{3}{4}$ for all strings x.

We are going to concatenate $t(|x|)$ witnesses $u_1, u_2, \ldots, u_{t(|x|)}$ to obtain a longer witness w. To simplify the notation we define the function $S(< x, w >) : \{0,1\}^* \to \mathbf{N}$ that counts those component witnesses that satisfy $< x, u_i > \in B$. Set

$$S(< x, w >) = \begin{cases} |\{i| < x, u_i > \in B\}| & \text{if } w = u_1 u_2 \ldots u_{t(|x|)}, |u_i| = s(|x|) \\ 0 & \text{if } w \text{ is not of this form} \end{cases} .$$

Let $A = \{< x, w > \mid S(< x, w >) > t(|x|)/2\}$. Intuitively, A concatenates t witnesses and uses a majority vote to decide membership. Set $q(n) = t(n)s(n)$, the length of the witnesses in the definition of A. The function q is polynomial since t and s are.

For x in L each $< x, u_i >$ is in B with μ^β-probability greater than $\frac{3}{4}$. Thus, by applying the Chernoff bound with $\varepsilon = \frac{1}{3}$ and $n = t(|x|)$, we have:

$$\begin{aligned} & \mu^\beta(\{w \in \{0,1\}^{q(|x|)} | < x, w > \in A\}) \\ = \; & Pr\left(S(< x, w >) > \frac{t(|x|)}{2}\right) \\ = \; & Pr\left(S(< x, w >) > (1 - \frac{1}{3}) \cdot t(|x|) \cdot \frac{3}{4}\right) \\ = \; & 1 - Pr\left(S(< x, w >) \leq (1 - \frac{1}{3}) \cdot t(|x|) \cdot \frac{3}{4}\right) \\ \geq \; & 1 - e^{-(\frac{1}{3})^2 \cdot t(|x|) \cdot \frac{3}{4} \cdot \frac{1}{2}} \\ \geq \; & 1 - 2^{-(\frac{1}{3})^2 \cdot t(|x|) \cdot \frac{3}{4} \cdot \frac{1}{2}} \\ \geq \; & 1 - 2^{-\frac{t(|x|)}{24}} \\ \geq \; & 1 - 2^{-\frac{t(|x|)}{48}} \\ = \; & 1 - 2^{-r(|x|)}. \end{aligned}$$

Similarly, for x not in L each $< x, u_i >$ is in B with μ^β-probability less than $\frac{1}{4}$ so we have:

$$
\begin{aligned}
&\mu^{\beta}(\{w \in \{0,1\}^{q(|x|)} \mid <x,w> \in A\}) \\
&= \; Pr\left(S(<x,w>) \leq \frac{t(|x|)}{2}\right) \\
&\geq \; Pr\left(S(<x,w>) < \frac{t(|x|)}{2}\right) \\
&= \; Pr\left(S(<x,w>) < (1+\frac{1}{2}) \cdot t(|x|) \cdot \frac{1}{4}\right) \\
&= \; 1 - Pr\left(S(<x,w>) \geq (1+\frac{1}{2}) \cdot t(|x|) \cdot \frac{1}{4}\right) \\
&\geq \; 1 - e^{-(\frac{1}{2})^2 \cdot t(|x|) \cdot \frac{1}{4} \cdot \frac{1}{3}} \\
&\geq \; 1 - 2^{-(\frac{1}{2})^2 \cdot t(|x|) \cdot \frac{1}{4} \cdot \frac{1}{3}} \\
&= \; 1 - 2^{-\frac{t(|x|)}{48}} \\
&= \; 1 - 2^{-r(|x|)}.
\end{aligned}
$$

□

Amplification lemmas such as the previous one tell us that by repeating membership trials a polynomial number of times we can make the error in probabilistic algorithms exponentially small. This strengthens our belief that probabilistic algorithms form the best model for feasible computation since for most real problems an exponentially small amount of error is acceptable.

3.1.1 The relationship between $\mathrm{BP}_{\beta}\mathrm{P}$ and BPP

The first question we ask and partially answer is: For which β does $\mathrm{BP}_{\beta}\mathrm{P} = \mathrm{BPP}$? We will show that for every β the inclusion $\mathrm{BPP} \subseteq \mathrm{BP}_{\beta}\mathrm{P}$ holds. In the other direction the inclusion depends on our ability to 'simulate' a coin with bias β using a fair coin. This is easily done with a rational bias or even a polynomial-time computable bias. We will extend this ability to BPP-*computable* biases.

The standard technique for simulating a fair coin using an unfair coin is due to von Neuman [77]. His method allows us to prove that for every $0 < \beta < 1$ we have $\mathrm{BPP} \subseteq \mathrm{BP}_\beta\mathrm{P}$. The technique involves working with pairs of coin tosses. We toss the unfair coin twice. If the tosses match we discard the result and flip again repeating as necessary until we get an unmatched pair of flips. When the coins do not match we use the first as our result. Since the probability of a 01 pair is $(1-\beta)\beta$ and the probability of a 10 pair is $\beta(1-\beta)$ we see that 01, yielding 0, and 10, yielding 1, are equally likely and we have simulated a fair coin. This idea underlies the following theorem.

Theorem 3.1.3.

If $0 < \beta < 1$ then $\mathrm{BPP} \subseteq \mathrm{BP}_\beta\mathrm{P}$.

Proof.

Let $L \in \mathrm{BPP}$, then there exist a language $A \in \mathrm{P}$ and a polynomial $q(n)$ with non-negative integer coefficients such that for every $x \in \{0,1\}^*$,

$$\mu\left(\{w \in \{0,1\}^{q(|x|)} | < x, w > \in A \iff x \in L\}\right) > \frac{7}{8}.$$

Set $\delta = 2\beta(1-\beta)$. Then δ is the probability that two bits, each with probability β of being 1, will not match. Chose a polynomial $r(n)$ with even coefficients that simultaneously satisfies $r(n) > 4q(n)/\delta$ and $r(n) > 16\ln(8)/\delta$. One such polynomial is $r(n) = 2\left(\left\lceil\frac{2}{\delta}\right\rceil q(n) + \left\lceil 8\frac{\ln(8)}{\delta}\right\rceil\right)$.

Next we define a transformation τ that will implement the simulation we described in the paragraph preceding this theorem. The intuition is that we consider the bits of a string of length $r(n)$ pairwise, discard any pairs that are either 00 or 11, and drop the second bit from any pair that is either 10 or 01 as given in the von Neumann simulation. We will subsequently use the first $q(n)$ bits of $\tau(w)$ if there are that many.

Define $\tau : \{0,1\}^{EVEN} \to \{0,1\}^*$ as follows

$$\tau(\lambda) = \lambda$$

$$\begin{aligned}
\tau(x00y) &= \tau(x)\tau(y) \\
\tau(x11y) &= \tau(x)\tau(y) \\
\tau(x10y) &= \tau(x)1\tau(y) \\
\tau(x01y) &= \tau(x)0\tau(y).
\end{aligned}$$

We now use τ to build a language $B \in \mathrm{P}$ to witness that $L \in \mathrm{BP}_\beta\mathrm{P}$.

Let

$$B = \{< x, w > \mid |w| = r(|x|), |\tau(w)| \geq q(|x|), < x, \tau(w)[0..q(|x|)-1] > \in A\}.$$

Then since the bits of w were chosen according to μ^β they are independent and for those w with $|\tau(w)| \geq q(|x|)$ we will have $\tau(w)[0..q(|x|)-1]$ uniformly distributed over $\{0,1\}^{q(|x|)}$.

Next we show that the set of w for which $\tau(w)$ has length less than $q(|x|)$ is small. We have chosen $r(|x|)$ to be at least $4q(|x|))/\delta$. Since the probability of a 'useful' pair of bits is δ, the expected length of $\tau(w)$ is at least $2q(|x|)$, far in excess of the $q(|x|)$ we need in the definition of B. The Chernoff bounds tells us that for w of length $r(n)$ we have

$$\begin{aligned}
Pr\left(|\tau(w)| < q(n)\right) &\leq Pr\left(|\tau(w)| < \frac{r(n)\delta}{4}\right) \\
&= Pr\left(|\tau(w)| < \left(1 - \frac{1}{2}\right)\frac{r(n)\delta}{2}\right) \\
&\leq e^{-\frac{\left(\frac{1}{2}\right)^2 \frac{r(n)}{2}\delta}{2}} \\
&\leq e^{-\frac{r(n)\delta}{16}} \\
&\leq e^{-\ln 8} \\
&= \frac{1}{8}.
\end{aligned}$$

Thus for every string x,

$$\begin{aligned}
&\mu^\beta\left(\{w\in\{0,1\}^{r(|x|)}|<x,w>\in B \iff x\in L\}\right)\\
&\geq \mu^\beta\left(\{w\in\{0,1\}^{r(|x|)}|\ |\tau(w)|\geq q(|x|),\right.\\
&\qquad\qquad <x,\tau(w)[0..q(|x|)-1]>\in A \iff x\in L\})\\
&= \mu^\beta\left(\{w\in\{0,1\}^{r(|x|)}|\ |\tau(w)|\geq q(|x|)\}\right)\\
&\quad \cdot\mu\left(\{w\in\{0,1\}^{q(|x|)}|<x,w>\in A \iff x\in L\}\right)\\
&\geq \frac{7}{8}\cdot\frac{7}{8}\\
&> \frac{3}{4}.
\end{aligned}$$

Hence, since $r(n)$ is a polynomial, and B is in P, we have $L \in \mathrm{BP}_\beta\mathrm{P}$. □

The reverse inclusion, $\mathrm{BP}_\beta\mathrm{P} \subseteq \mathrm{BPP}$ fails for some β. Intuitively one reason this is true is because we can encode information in the bias and extract it during the computation. In fact, we can compute non-recursive languages using the information coded in the bias. To show this we will relate $\mathrm{BP}_\beta\mathrm{P}$ to advice classes.

Definition.

If $\mathcal{C}$ is a class of languages, $\mathcal{F}$ a class of functions mapping $\mathbf{N}$ into $\{0,1\}^*$and L is a language, then we say $L \in \mathcal{C}/\mathcal{F}$ if there exist a language $A \in \mathcal{C}$ and a function $h \in \mathcal{F}$ such that $L = \{x| <x, h(|x|)> \in A\}$.

We call $\mathcal{C}/\mathcal{F}$ an *advice class* and call h an *advice function* for L.

We will use the function classes CLog and Poly defined next, to form the advice classes P/CLog and P/Poly respectively. (Note: The class P/CLog has also recently been studied by Hermo [34], who calls it Pref-P/ log. Her work cites Ko [40] on the related class STRONG-P/ log which she calls Full-P/ log.) P/CLog and P/Poly will provide upper and lower bounds for the $\mathrm{BP}_\beta\mathrm{P}$ classes. The *cumulative log* functions in CLog are the functions mapping integers to strings that are cumulative, *i.e.,* $f(n) \sqsubseteq f(n+1)$, and whose values are $O(\log n)$ in length. CLog provides us with a class of advice functions that grow very slowly in length. The growth is slow enough to allow us to use the Chernoff bounds to guarantee that flipping coins can extract all of the information encoded in the advice string.

Definition.

A function $h : \mathbf{N} \to \{0,1\}^*$ is in the class CLog if it satisfies

(i) $h(n) \sqsubseteq h(n+1)$ and

(ii) $|h(n)| = O(\log n)$.

A function $h : \mathbf{N} \to \{0,1\}^*$ is in the class $Poly$ if it satisfies $|h(n)| = O(n^k)$ for some k.

Advice classes such as P/Poly and P/CLog are uncountable and therefore contain non-recursive languages. The next theorem says that with the right bias, we can compute some of these non-recursive languages using $\mathrm{BP}_\beta\mathrm{P}$ algorithms.

The following technical lemma is a specialization of the Chernoff bounds adapted to our needs in the theorem that it precedes.

Lemma 3.1.4.

If $\frac{15}{27} < \beta < \frac{19}{27}$, $n > 6 \cdot 3^{2k}$, and ψ is the proportion of successes in n independent trials of a Bernoulli experiment each trial having probability β of success, then

$$Pr\left(|\beta - \psi| > 3^{-k}\right) < \frac{1}{4}.$$

Equivalently, if ψ is the proportion of 1's in string z, then

$$\mu^\beta\left(\left\{z \in \{0,1\}^{n(k)} \,|\, |\beta - \psi| < 3^{-k}\right\}\right) > \frac{3}{4}.$$

Proof.

The Chernoff bound with $S = \psi n$ gives us

$$Pr\left(|\psi - \beta\ | > \varepsilon\beta\right) < 2e^{-\frac{\varepsilon^2 \beta n}{3}}.$$

By hypothesis $\frac{15}{27} < \beta < \frac{19}{27}$, thus,

$$Pr\left(|\psi - \beta\ | > \varepsilon\frac{19}{27}\right) < 2e^{-\frac{\varepsilon^2 15n}{3\cdot 27}}.$$

Letting $3^{-k} = \frac{19}{27}\varepsilon$ yields

$$Pr\left(|\psi - \beta\ | > 3^{-k}\right) < 2e^{-\frac{3^{-2k}\cdot 27^2\cdot 15n}{19^2\cdot 3\cdot 27}}.$$

Since $n > 6 \cdot 3^{2k}$,

$$2e^{-\frac{3^{-2k}\cdot 27^2\cdot 15n}{19^2\cdot 3\cdot 27}} < 2e^{-\frac{27^2\cdot 15\cdot 6}{19^2\cdot 3\cdot 27}} < \frac{1}{4}.$$

□

<u>Theorem 3.1.5.</u>

If $L \in$ P/CLog then there exists $0 < \beta < 1$, such that $L \in \mathrm{BP}_\beta\mathrm{P}$.

<u>Proof.</u>

Let $L \in$ P/CLog be given. Then there exist a language $A \in P$, a constant $c \in \mathbf{N}$, and a CLog advice function $h : \mathbf{N} \to \{0,1\}^*$ that satisfy $|h(n)| \le c \cdot \log n$, and $L = \{x| < x, h(|x|) >\in A\}$.

Since $h(n) \sqsubseteq h(n+1)$, the sequence $\hat{z} = \lim_{n\to\infty} h(n)$ is well defined.

Define $f : \{0,1\}^\infty \to \mathbf{R}$ by $f(z) = \frac{5}{9} + \frac{1}{27}\sum_{j=0}^{\infty} 2z[j] \cdot 3^{-j}$. This yields a real number between 5/9 and 2/3 whose ternary expansion begins with 0.12 and that contains no other 1s. We use a ternary encoding without 1s to obtain a totally disconnected set of reals and to make f 1-1. With the standard binary interpretation of sequences as reals, for example, 1010011111... and 101010000... would both map to the same real number. Our representation avoids this complication.

Set $\beta = f(\hat{z})$. We shall show that $L \in \mathrm{BP}_\beta\mathrm{P}$.

The function f is 1-1. Further, it is the case that for any $r_1, r_2 \in$ range(f) with $|r_1 - r_2| < 3^{-(k+2)}$, $f^{-1}(r_1)$ and $f^{-1}(r_2)$ agree in their first k bits.

For an input x of length m we need to recover the initial $k = c \log m$ bits of $\hat{z}$. By the previous lemma and the comment above about f, we need a μ^β-random string of length at least $6 \cdot 3^{2k}$ to obtain a high probability of achieving this. Since $6 \cdot 3^{2k} = 6m^{2c \log 3}$, we can use $n(k) = 6m^{4c}$ which is polynomial in m as needed for $\mathrm{BP}_\beta\mathrm{P}$ witness lengths.

Let x be an input string. As described above let $m = |x|$, $k(m) = |h(m)|$, and $n(m) = 6m^{4c}$.

Define $\psi : \{0,1\}^* \times \{0,1\}^* \to \mathbf{R}$ as follows. When the first argument to $\psi(x,z)$ is of length m, the second has length $n(m)$. The value of ψ is given by $\frac{\#(1,z)}{|z|}$ truncated to $k(m)$ digits in ternary expansion using ...022222... rather than ...100000... if it occurs.

Intuitively, we are using $n(m)$ bits each with probability β of being 1. The function ψ gives the proportion of 1s among these bits. By the Chernoff bound, $\psi(x,z)$ is very close to β with high probability, in which case f^{-1} converts the ternary form of $\psi(x,z)$ to a sequence that agrees (exactly) with $\hat{z}$ in its first $k(m)$ bits. The bits of $\hat{z}$ yield those of h so we now have recovered h from the β-biased random bits.

More precisely, for $\psi(x,z) \in range(f)$, set $\hat{h}(z)$ equal to the first k bits of $f^{-1}(\psi(x,z))$ and set $\hat{h}(z) = \lambda$ otherwise.

Let $\hat{A} = \{< x, z > \mid < x, \hat{h}(z) > \in A\}$. Since $A \in P$ and $\hat{h}$ is polynomial-time computable, we also have that $\hat{A} \in P$. By the previous lemma we have

$$Pr_{\mu^\beta}(|\psi(x,z) - \beta| < 3^{-(k(m)+2)}) > \frac{3}{4}.$$

When the inequality $|\psi(x,z) - \beta| < 3^{-(k(m)+2)}$ holds, the first $k(m)$ bits of $\hat{h}$ equal $h(m)$ so it follows that for each x

$$Pr_{\mu^\beta}(< x, z > \in \hat{A} \iff x \in L) > \frac{3}{4}.$$

Hence $L \in \mathrm{BP}_\beta\mathrm{P}$. □

Reiterating what we said at the beginning of this chapter, this theorem shows that there are βs for which $\mathrm{BP}_\beta\mathrm{P}$ contains non-computable languages. Since the languages of BPP are computable, this is sufficient to show that for these β we must have BPP $\neq$ BP$_\beta$P. Furthermore, it is not hard to see that such β are actually dense in the unit interval. In particular, this means that arbitrarily close to $\frac{1}{2}$ lie real biases with this computational power. At first glance then it would seem that BPP is extremely sensitive to the "fairness" of the coin. It would seem that any deviation, such as might arise in a "real-world" source of randomness, would cause BPP to break. We shall examine this fragility in the last section of this chapter and discuss the sense in which the class BPP actually can be made quite robust against variation in the probability distribution.

Returning to the line of thought that lead to the previous theorem, advice classes also provide an upper bound for the complexity of BP$_\beta$P, as the following theorem shows.

Theorem 3.1.6.

For all $0 < \beta < 1$, $\quad$ BP$_\beta$P $\subseteq$ P/Poly.

Proof.

Let $0 < \beta < 1$ and let $L \in$ BP$_\beta$P. Let A and q be as given by Lemma 3.1.2 applied to L with $r(n) = 2n + 2$.

Let $\Gamma_x = \{\gamma \in \{0,1\}^\infty | < x, \gamma[0..q(|x|) - 1] > \in A \iff x \notin L\}$. By the lemma 3.1.2, $\mu^\beta(\Gamma_x) \leq 2^{-(2|x|+2)}$. If we let $\Gamma = \cup_x \Gamma_x$, then Γ is the set of infinite sequences γ, for which we can find an x such that $\gamma[0..q(|x|) - 1]$ fails to serve as a witness for x. We now have

$$\begin{aligned}
\mu^\beta(\Gamma) &\leq \sum_{x \in \{0,1\}^*} 2^{-(2|x|+2)} \\
&= \sum_{n \in \mathbf{N}} \sum_{x \in \{0,1\}^n} 2^{-(2|x|+2)} \\
&= \sum_{n \in \mathbf{N}} 2^n 2^{-(2n+2)} \\
&= \sum_{n \in \mathbf{N}} 2^{-n-2} \\
&= \frac{1}{2}
\end{aligned}$$

In particular, $\Gamma^c \neq \phi$, so some infinite sequence γ has initial segments that are valid witnesses for every string. Choose $\gamma \in \Gamma^c$. Then for every $x \in \{0,1\}^*$ we have that $< x, \gamma[0..q(|x|)-1] > \in A \iff w \in L$. From γ we can define an advice function $h(i) = \gamma[0..q(i)-1]$ that 'works' for every x. Thus, since h is polynomial-time computable and $A \in \mathrm{P}$, we have shown that $L \in \mathrm{P/Poly}$. $\square$

Versions of the previous two theorems were proven by Kilian and Siegelmann [69] in the context of probabilistic neural networks with arbitrary real weights.

To further answer the "When does BPP = $\mathrm{BP}_\beta\mathrm{P}$?" we want to use some resource bounded measure results [20]. This will apply to the current topic through almost classes.

Investigation of the connection between almost classes and BPP dates to work by Bennett and Gill [10] and Ambos-Spies [1]. We will extend their results to $\mathrm{BP}_\beta\mathrm{P}$ and use that to help answer our question. First we give a definition of almost$_\nu$-P and the corresponding BPP result.

Definition.

Given a language $L \subseteq \{0,1\}^*$ and a probability measure ν on $\{0,1\}^\infty$, L is said to be in the class almost$_\nu$-P if $\nu\left(\{A \in \{0,1\}^* | L \leq_{\mathrm{T}}^{\mathrm{P}} A\}\right) = 1$.

We write almost-P for almost$_\mu$-P, and almost$_\beta$-P for almost$_{\mu^\beta}$-P.

Bennett and Gill [10] and Ambos-Spies [1] proved that BPP = almost-P. We will extend their proof to show that $\mathrm{BP}_\beta\mathrm{P}$ = almost$_\beta$-P. First we prove a version of the Lebesque Density Theorem that will be useful for this purpose.

Theorem 3.1.7 (Lebesque Density Theorem for $\{0,1\}^\infty$).

Let , ν be a measure on $\{0,1\}^\infty$, $S \subset \{0,1\}^\infty$ be a set of positive $\nu-$measure, and $\varphi(S) = \{x \in S | \lim_{k\to\infty} \frac{\nu(S\cap C_{x[0\ldots k]})}{\nu(C_{x[0\ldots k]})} = 1\}$. (The points of $\varphi(S)$ are called the *density points* of S.) Then $\nu(S - \varphi(S)) = 0$. In particular the set $\varphi(S)$ is non-empty; in fact almost every point of S is a density point of S.

Proof.

Let $A_n = \{x \in S | \liminf_{k\to\infty} \frac{\nu(S\cap C_{x[0...k]})}{\nu(C_{x[0...k]})} < 1 - \frac{1}{n}\}$. Then $S - \varphi(S) = \cup_{n=1}^{\infty} A_n$ and it is sufficient to show that $\nu(A_n) = 0$ for all n.

Suppose not, then there is an n such that $\nu^*(A_n) > 0$, where ν^* is the outer measure associated with ν. *i.e.,* $\nu^*(T) = glb(\{\nu(G) | T \subseteq G \text{ and } G \text{ is open }\})$. Let G be an open set containing A_n and satisfying $\nu^*(A_n) \le \nu(G) \le \frac{n}{n-1}\nu^*(A_n)$.

Define $\mathcal{E} = \{x \in \{0,1\}^* | (i) C_x \subseteq G, (ii) \frac{\nu(S\cap C_x)}{\nu(C_x)} < 1 - \frac{1}{n}$ and

(iii) no proper prefix of x satisfies (i) and (ii)$\}$.

Note that for $x_1, x_2 \in \mathcal{E}$ with $x_1 \ne x_2$ we have $C_{x_1} \cap C_{x_2} = \phi$. Thus we have

$$\begin{aligned} \nu^*(A_n \cap \bigcup_{x\in\mathcal{E}} C_x) &\le \sum_{x\in\mathcal{E}} \nu(S\cap C_x) \\ &\le (1-\frac{1}{n}) \sum_{x\in\mathcal{E}} \nu(C_x) \\ &\le (1-\frac{1}{n})\nu(G) \\ &< \nu^*(A_n). \end{aligned}$$

This shows that $\nu^*(A_n - \bigcup_{x\in\mathcal{E}} C_x) > 0$ and thus $A_n - \bigcup_{x\in\mathcal{E}} C_x \ne \phi$ and we may fix an $x_0 \in A_n - \bigcup_{x\in\mathcal{E}} C_x$. Since x_0 is in A_n we have that there exist $k_1 < k_2 < \ldots$ such that $\frac{\nu(S\cap C_{x_0[0...k_j]})}{\nu(C_{x_0[0...k_j]})} < 1 - \frac{1}{n}$.

Since G is open there is a k_j such that $C_{x_0[0...k_j]} \subseteq G$. Thus some prefix $\hat{x}$ of $x_0[0\ldots k_j]$ must be in $\mathcal{E}$. But then $x_0 \in \bigcup_{x\in\mathcal{E}} C_x$ in contradiction to the way x_0 was selected. Therefore our assumption that $\nu^*(A_n) > 0$ must be in error and we have shown that for every n we have $\nu^*(A_n) = 0$ and thus $\nu(S - \varphi(S)) = 0$. $\square$

Theorem 3.1.8.

$\mathrm{BP}_\beta\mathrm{P} = \mathrm{almost}_\beta\text{-P}$.

Proof.

($\supseteq$) Following Ambos-Spies's proof [1] we show that if $\mu^{\beta}(P^{-1}(L)) > 0$ then $L \in \mathrm{BP}_{\beta}\mathrm{P}$.

Suppose that $L \subseteq \{0,1\}^*$ is given satisfying $\mu^{\beta}(P^{-1}(L)) > 0$. If $M_0, M_1, M_2, \ldots$ is an enumeration of polynomial time-bounded, oracle Turing machines, then $P^{-1}(L) = \bigcup_{i \in \mathbf{N}} \{A | L = L(M_i^A)\}$. This is a countable union with positive measure, so there must be a single polynomial time-bounded, oracle Turing machine M, such that $\mu^{\beta}(\{A | L = L(M^A)\}) > 0$. Let $q(n)$ be a polynomial time bound for M. By the Lebesque Density Theorem, there must be an oracle prefix y such that $\frac{\mu^{\beta}(\{B | y \sqsubseteq B \text{ and } L = L(M^B)\})}{\mu^{\beta}(\{B | y \sqsubseteq B\})} > \frac{3}{4}$.

We now modify M to produce a (non-oracle) Turing machine accepting a language witnessing that $L \in \mathrm{BP}_{\beta}\mathrm{P}$. Let $\hat{M}$ simulate M as follows. The input to $\hat{M}$ is of the form $< x, w >$ where x is the input to M and w is a random witness string. $\hat{M}$ has y hardwired into its control. Any time $M(x)$ queries the string s_i with $i < |y|$ then $\hat{M}(< x, w >)$ looks at bit $y[i]$ to simulate the response to the query. The j-th time $M(x)$ queries the string s_i with $i \geq |y|$, $\hat{M}(< x, w >)$ uses bit $w[j]$ of the witness string to answer the query. $\hat{M}$ must remember its queries and answers so that if a query s_i with $i \geq |y|$ is repeated it receives the same answer. (Or, without loss of generality, assume that no queries are repeated.)

Then for all $x \in \{0,1\}^*$, we have

$$\mu^{\beta}(\{w \in \{0,1\}^{q(|x|)} | < x, w > \in L(\hat{M}) \iff x \in L\}) > \frac{3}{4}.$$

Thus, $L(\hat{M})$ and $q(n)$ witness that $L \in \mathrm{BP}_{\beta}\mathrm{P}$ and we have shown that almost$_{\beta}$-P $\subseteq \mathrm{BP}_{\beta}\mathrm{P}$.

($\subseteq$) Let $L \in \mathrm{BP}_{\beta}\mathrm{P}$. We will show that $\mu^{\beta}\left(\{B \subseteq \{0,1\}^* | L \not\leq_{\mathrm{T}}^{\mathrm{P}} B\}\right) = 0$. Let $\delta > 0$ be given. Let $r(n) = 2n + 1 - \log \delta$. Let A and $q(n)$ be as given by Lemma 3.1.2 applied to L and r, *i.e.,* for each string x,

$$\mu^{\beta}\left(\{w \in \{0,1\}^{q(|x|)} | < x, w > \in A \iff x \in L\}\right) > 1 - 2^{-r(|x|)}.$$

Let M be a polynomial time-bounded Turing machine accepting A. From M we define an oracle Turing machine $\hat{M}$ as follows. When $M(< x, w >)$ would refer to witness bit $w[i]$, $\hat{M}^B(x)$ queries the oracle for string $x0^i$.

This establishes a one-to-one measure-preserving correspondence between the witness strings $w \in \{0,1\}^{q(|x|)}$ and a set of $2^{|w|}$ generalized cylinders $C_{\phi(w)}$ whose values at certain widely spaced bits are equal to those of w. These satisfy $< x, w > \in A \iff M^B(x)$ accepts for every oracle B in $C_{\phi(w)}$. Hence, we have

$$\mu^\beta\left(\{B \subseteq \{0,1\}^* | \hat{M}^B(x) \text{ accepts }\}\right) = \mu^\beta\left(\{w \in \{0,1\}^{q(|x|)} | < x, w > \in A\}\right).$$

Using the complements of these sets, for each x we have

$$\begin{aligned}
\mu^\beta\left(\{B | L \not\leq_{\mathrm{T}}^{\mathrm{P}} B\}\right) &\leq \mu^\beta\left(\{B | \hat{M}^B(x) \neq x \in L \text{ for some } x\}\right) \\
&= \mu^\beta\left(\bigcup_{x \in \{0,1\}^*} \{B | \hat{M}^B(x) \neq x \in L\}\right) \\
&\leq \sum_{x \in \{0,1\}^*} \mu^\beta\left(\{B | \hat{M}^B(x) \neq x \in L\}\right) \\
&\leq \sum_{x \in \{0,1\}^*} 2^{-(2|x|+1-\log\delta)} \\
&= \sum_{n \in \mathbf{N}} \sum_{|x|=n} 2^{-(2|x|+1-\log\delta)} \\
&= \sum_{n \in \mathbf{N}} 2^n 2^{-(2n+1-\log\delta)} \\
&= \delta.
\end{aligned}$$

The value of δ was arbitrary so $\mu^\beta\left(\{B | L \not\leq_{\mathrm{T}}^{\mathrm{P}} B\}\right) = 0$ and we have shown that $L \in$ almost$_\beta$-P, and hence BP$_\beta$P = almost$_\beta$-P. □

We continue with the question: For which β does BPP = BP$_\beta$P? We see that if approximations to β are polynomial-time computable in the sense of Ko [41], then

$\mathrm{BP}_\beta\mathrm{P} = \mathrm{BPP}$. This is true because we can first compute an approximation to β and then use some of the bits of the randomly chosen witness string to simulate a μ^β-random bit. The set of witnesses that do not 'work' can be hidden in a measure-zero set yielding almost$_\beta$-P $\subseteq$ BPP and hence BPP $= \mathrm{BP}_\beta\mathrm{P}$ for the p-computable β's.

The theorems that follow use a naive technique for approximately simulating a β-biased coin from a fair coin and an approximation for β. Given $\hat{\beta}$, an n-bit approximation to β, one flips n coins and interprets their concatenation as a real number between zero and one. A real number chosen uniformly at random from $[0,1]$ is less than or equal to $\hat{\beta}$ with probability $\hat{\beta}$. Thus we can use n fair bits to simulate a single bit that has a bias approximately equal to β.

The following theorem can be proven directly using the above argument. It is also a corollary to Theorem 3.1.11 below or Theorem 4.6.1 in the next chapter. We state it here to give a smooth progression through p-computable βs to the probabilistically computable βs we introduce below.

Theorem 3.1.9.

If $0 < \beta < 1$ and $\beta \in \mathrm{P_{CF}}$ then $\mathrm{BP}_\beta\mathrm{P} = \mathrm{BPP}$.

Proof.

This theorem will follow either from Theorem 3.1.11 or from Theorem 4.6.1. $\square$

The biases in the previous theorem are polynomial-time computable in a deterministic sense. That is, the probabilistic computation computes the approximations to β deterministically and uses its probabilistic power only in computing the language after it has β. It actually has more power at its disposal. It could use its probabilistic power in computing the approximations as well. We next define precisely what it means for a real number to be BPP-computable.

Definition.

We say that a real number x is BPP-*computable* if there exist a polynomial-time computable function $\varphi : \{0\}^* \times \{0,1\}^* \to \mathbf{D}$ and a polynomial p satisfying

$$\mu\left(\{w \in \{0,1\}^{p(n)} |\ |\varphi(0^n, w) - x| < 2^{-n}\}\right) > \frac{3}{4}$$

for every $n \in \mathbf{N}$. We define $\mathrm{BPP_{CF}}$ to be the class of all such x and for each x we define the class $\mathrm{BPCF}(x)$ to be the class of all corresponding functions.

This definition extends naturally to $\mathrm{BP_{\beta}P}$-computable reals. We will not explore the properties of those reals in this thesis.

As is typical with BPP-type definitions, the $\frac{3}{4}$ may be replaced with any constant lying strictly between $\frac{1}{2}$ and 1. Furthermore, as usual, we can amplify the probability to be exponentially close to one via the following lemma.

Lemma 3.1.10 ($\mathrm{BPP_{CF}}$ amplification).

Let $\beta \in \mathrm{BPP_{CF}}$, then for every polynomial r, there exist a $\psi \in \mathrm{BPCF}(x)$ and a polynomial q such that $\mu\left(\{w \in \{0,1\}^{p(n)} |\ |\varphi(0^n, w) - x| < 2^{-n}\}\right) > 1 - 2^{-r(n)}$.

Proof.

Let $\varphi \in \mathrm{BPCF}(\beta)$ and polynomial p witness that $\beta \in \mathrm{BPP_{CF}}$. *i.e.*, $\mu\left(\{w \in \{0,1\}^{p(n)} |\ |\varphi(0^n, w) - x| < 2^{-n}\}\right) > \frac{3}{4}$ for every $n \in \mathbf{N}$. Let a polynomial r be given. Let $k(n) = 48r(n)$ and let $q(n) = (2k(n) + 1)p(n)$.

Define $\psi(0^n, w)$ to be the median of the values:

$$\{\varphi(0^n, w[jp(n)..(j+1)p(n) - 1]) | 0 \leq j \leq 2k(n)\}.$$

That is, compute $2k(n)+1$ approximations to β using φ and define ψ to be the median of this set of values. This process requires $q(n)$ witness bits since each computation of φ requires $p(n)$ such bits.

Since each approximation has probability at least $\frac{3}{4}$ of satisfying $|\varphi(0^n, w) - x| < 2^{-n}$, the Chernoff bounds tells us that the median has probability at least $1 - e^{-(2k(n)+1)/96}$ of satisfying the same inequality.

Our choice of k and q yields

$$\mu\left(\{w \in \{0,1\}^{q(n)} | \ \ |\psi(0^n, w) - \beta| < 2^{-n}\}\right) \geq 1 - e^{-(2k(n)+1)/96} \geq 1 - 2^{r(n)}$$

and the lemma is proven. □

If β is a BPP-computable real then there is a BPP computation that can first approximate β and then simulate a $\mathrm{BP}_\beta\mathrm{P}$ computation. Hence we have the following theorem.

Theorem 3.1.11.

If $0 < \beta < 1$ and $\beta \in \mathrm{BPP}_{\mathrm{CF}}$ then almost$_\beta$-P $\subseteq$ almost-P, so $\mathrm{BP}_\beta\mathrm{P} = \mathrm{BPP}$.

Proof.

Let $0 < \varepsilon < 1$ be given. Set $r(k) = 2k - log\varepsilon + 3$. Let $\beta \in \mathrm{BPP}_{\mathrm{CF}}$ be given, and let ψ and q be given by the previous lemma applied to β and r,*i.e.*, $Pr(|\psi(0^k, w) - \beta| < 2^{-k}) > 1 - 2^{-r(n)}$ when w is chosen uniformly at random from $\{0,1\}^{q(k)}$. For $i \in \mathbf{N}$ and $A \in \{0,1\}^\infty$ we define

$\xi(A) = [\![0 \in A]\!][\![00 \in A]\!]...$

$\tau_i(A) = \sum_{j=1}^{\infty} [\![s_i 10^j \in A]\!] \cdot 2^{-j}$.

The BPCF(β) function ψ will use initial segments of ξ to compute its approximations to β. The value τ_i will be used to simulate the i-th bit of a biased oracle using a fair oracle since $Pr[\tau_i(A) \leq \beta] = \beta$ if A is a randomly selected unbiased oracle.

Based on the τ_i's we define some ordinary and special cylinders

$C_{i,0} = \{A | A[i] = 0\}$ and $C_{i,1} = \{A | A[i] = 1\}$, and

$D_{i,1} = \{A | \tau_i(A) \leq \beta\}$ and $D_{i,0} = \{x | \tau_i(A) > \beta\}$.

Observe that $\mu(D_{i,1}) = \beta = \mu^\beta(C_{i,1})$ and that $\mu(D_{i,0}) = 1 - \beta = \mu^\beta(C_{i,0})$. Let $\hat{D}_i = \{A \mid |\psi(0^{r(|s_i|)}, \xi(A)^{\leq q(r(|s_i|))}) - \tau_i(A)^{\leq r(|s_i|)}| \leq 2^{-r(|s_i|)}\}$ and,
$\hat{E}_i = \{A \mid |\psi(0^{r(|s_i|)}, \xi(A)^{q(r(|s_i|))}) - \beta| > 2^{-r(|s_i|)}\}$.

The $\hat{E}$'s contain the oracles that do not allow ψ to compute an accurate approximation to β, while the $\hat{D}$'s contain the oracles that produce τ_i's too close to β to accurately determine which is smaller. By our choice of ψ and q, we have $\mu(\hat{E}_i) < 2^{-r(|s_i|)}$. Further, since an interval of width $2 \cdot 2^{r(|s_i|)}$ can contain at most three dyadic rationals with denominator $2^{r(|s_i|)}$, we know that $\mu(\hat{D}_i) \leq 3 \cdot 2^{-r(|s_i|)}$.

Let $X = \left(\bigcup_{i\in\mathbf{N}} \hat{D}_i\right) \cup \left(\bigcup_{i\in\mathbf{N}} \hat{E}_i\right)$. Then

$$\begin{aligned}
\mu(X) &\leq \sum_{i\in\mathbf{N}} \mu(\hat{D}_i) + \sum_{i\in\mathbf{N}} \mu(\hat{E}_i) \\
&\leq 3\sum_{i\in\mathbf{N}} 2^{-r(|s_i|)} + \sum_{i\in\mathbf{N}} 2^{-r(|s_i|)} \\
&\leq 4\sum_{i\in\mathbf{N}} 2^{-r(|s_i|)} \\
&= 4\sum_{k\in\mathbf{N}} 2^k 2^{-r(k)} \\
&= 4\sum_{k\in\mathbf{N}} 2^k 2^{-2k+log\varepsilon-3} \\
&= \varepsilon.
\end{aligned}$$

So we know that the μ measure of X is small. To show that the μ^β measure is also small we define a measure preserving mapping as follows.

Define $\varphi : \{0,1\}^\infty \to \{0,1\}^\infty$ by

$$\varphi(w)[i] = \begin{cases} 1 & w \in D_{i,1} \\ 0 & w \in D_{i,0}. \end{cases}$$

Then $\varphi(D_{i,0}) = C_{i,0}$ and $\varphi(D_{i,1}) = C_{i,1}$. Also, $\mu(\varphi^{-1}(C_{i,1})) = \mu(D_{i,1}) = \beta = \mu^\beta(C_{i,1})$. Similarly, $\mu(\varphi^{-1}(C_{i,0})) = \mu(D_{i,0}) = 1 - \beta = \mu^\beta(C_{i,0})$. In fact, since μ^β and $\mu \circ \varphi^{-1}$ are measures that agree on cylinders, by the uniqueness of measure extensions [33], we have that $\mu^\beta(S) = \mu(\varphi^{-1}(S))$ for all measurable sets. Since for any language L, $P^{-1}(L)$ is measurable, we have $\mu^\beta(P^{-1}(L)) = \mu(\varphi^{-1}(P^{-1}(L)))$. We will use this relationship later in this proof.

Having shown that the set of μ-random oracles that either cannot approximate β well enough or produce a random value too close to β is small, we now describe the simulation using a μ-random oracle in place of a μ^β-random oracle.

Let $L \in \text{almost}_\beta\text{-P}$ and $A \in P^{-1}(L)$. Then there is a p-OTM M that accepts L given oracle A. From M we define a new p-OTM M' that simulates M as follows. Whenever M queries s_i M' performs the following steps

1) compute $\hat{\xi} = \xi(A)^{\leq q(r(|s_i|))}$

2) compute $\hat{\psi} = \psi(0^{r(|s_i|)}, \hat{\xi})$

3) compute $\hat{\tau} = \tau_i(A)^{\leq r(|s_i|)}$

4a) if $\hat{\psi} - \hat{\tau} > 2^{-r(|s_i|)}$ then continue the simulation of M as if the query had been answered 1

4b) if $\hat{\tau} - \hat{\psi} > 2^{-r(|s_i|)}$ then continue the simulation of M as if the query had been answered 0

4c) if neither (4a) nor (4b) hold then we do not care what M' does since the oracle will be in X.

One sees that if $B = \varphi^{-1}(A)$ and $B \notin X$ then $L(M^A) = L(M'^B)$. So M' is a witness that $\varphi^{-1}(A) \in P^{-1}(L)$. This shows that $\varphi^{-1}(P^{-1}(L)) \subseteq P^{-1}(L)$. Finally then

$$\begin{aligned} \mu(P^{-1}(L)) &\geq \mu(\varphi^{-1}(P^{-1}(L))) \\ &\geq \mu(\varphi^{-1}(P^{-1}(L))) - \mu(X) \\ &= \mu^\beta(P^{-1}(L)) - \mu(X) \\ &\geq 1 - \varepsilon. \end{aligned}$$

But ε was arbitrary so $\mu(P^{-1}(L)) = 1$ and $L \in$ almost-P. Thus $\text{almost}_\beta\text{-P} \subseteq$ almost-P. The reverse inclusion is given by Theorem 3.1.3. The theorem then follows from Theorem 3.1.8. $\square$

We now have that $\mathrm{BP}_\beta\mathrm{P} = \mathrm{BPP}$ for BPP-computable βs. From the other direction we know that there are βs for which $\mathrm{BP}_\beta\mathrm{P} \neq \mathrm{BPP}$. We leave tightening this gap for future work.

3.1.2 $\mathrm{BP}_\beta\mathrm{P}$ and the relativized polynomial-time hierarchy

In this section we wish to remark on the place of $\mathrm{BP}_\beta\mathrm{P}$ in the relativized polynomial-time hierarchy. We do this analogously to the placement of BPP in the unrelativized polynomial-time hierarchy.

Gacs and Sipser [71] have shown that BPP lives in the second level of the polynomial-time hierarchy, *i.e.* $\mathrm{BPP} \subseteq \Sigma_2^{\mathrm{P}}$. Correspondingly, we will show that $\mathrm{BP}_\beta\mathrm{P} \subseteq \Sigma_2^{\mathrm{P}} f_\beta$ for any function oracle f_β in $CF(\beta)$.

Theorems 3.1.9 and 3.1.11 can be restated in terms of function oracles to yield the following theorem.

Theorem 3.1.12.

If $f_\beta \in \mathrm{CF}(\beta)$ then $\mathrm{BP}_\beta\mathrm{P}$ is contained in the function-oracle class $\mathrm{BPP}(f_\beta)$.

Proof.

The proof of this theorem is similar to that of Theorem 3.1.11. The algorithm queries the oracle for the approximation to β rather than computing it. Afterwards it continues as the algorithm in Theorem 3.1.11. □

The complexity of the class $\mathrm{BPP}(f_\beta)$ in the previous theorem is directly related to the computational complexity of β. If β is BPP-computable than the oracle is redundant as we can simulate it by probabilistically computing approximations to β.

Theorem 3.1.13.

If $\beta \in \mathrm{BPP}_{\mathrm{CF}}$ then $\mathrm{BPP}(f_\beta) = \mathrm{BPP}$.

Proof.

Since $\beta \in \mathrm{BPP}_{\mathrm{CF}}$ any algorithm can replace queries to the oracle with polynomial-time computations of the approximations to β. $\square$

The next corollary is a restatement of Theorem 3.1.11. It follows directly from the previous two theorems. It indicates that an alternative approach to the work of the preceding sections of this chapter is via function oracle for the biases.

Corollary 3.1.14.

If $\beta \in \mathrm{BPP}_{\mathrm{CF}}$ then $\mathrm{BP}_\beta\mathrm{P}$=BPP.

Proof.

Theorem 3.1.12 says that $\mathrm{BP}_\beta\mathrm{P} \subseteq \mathrm{BPP}(f_\beta)$. Theorem 3.1.13 says that $\mathrm{BPP}(f_\beta) \subseteq$ BPP. Since BPP $\subseteq \mathrm{BP}_\beta\mathrm{P}$ holds for all $0 < \beta < 1$ we have $\mathrm{BP}_\beta\mathrm{P} = \mathrm{BPP}$. $\square$

Gacs and Sipser's [71]proof that BPP $\in \Sigma_2^{\mathrm{P}}$ relativizes to $\mathrm{BPP}(f) \in \Sigma_2^{\mathrm{P}}(f)$ for any function oracle f. Thus we have the following theorem.

Theorem 3.1.15.

If f_β is any Cauchy function for β then $\mathrm{BP}_\beta\mathrm{P} \subseteq \Sigma_2^{\mathrm{P}}(f_\beta)$.

Proof.

By Theorem 3.1.12 $\mathrm{BP}_\beta\mathrm{P} \subseteq \mathrm{BPP}(f_\beta)$ and by Gacs and Sipser $\mathrm{BPP}(f_\beta) \subseteq \Sigma_2^{\mathrm{P}}(f_\beta)$. $\square$

This completes our exploration of the $\mathrm{BP}_\beta\mathrm{P}$ classes in this thesis. We now turn our attention to some of the implications of this work regarding the robustness of the class BPP.

3.2 BPP and Robustness

Since any real-world source of randomness is unlikely to be perfectly consistent, our view that BPP captures the essence of feasible computation is in jeopardy. In section

3.1.1 we illustrated this by showing that there is a dense set of biases that correspond to coins computing non-recursive languages. Hence even a little bit of variability in the distribution can force a BPP or $\mathrm{BP}_\beta\mathrm{P}$ algorithm to compute incorrect results.

The goal of this section is to address this lack of robustness and to show in what sense the standard model of BPP still captures the intended class of feasible languages.

Others have attacked this robustness question for BPP via semi-random sources [59, 76, 75], PRB-random sources [23], δ-random sources [82], and more recently extractors and dispersers [81, 82, 53]. We look at the robustness problem using the $\mathrm{BP}_\beta\mathrm{P}$ classes.

Our idea of a robust version of $\mathrm{BP}_\beta\mathrm{P}$ is that it should be insensitive to small fluctuations in the bias of the source of random bits. In our terms we mean that if the probability measure ν is 'close' to μ^β then both ν and μ^β should compute the same language from a given algorithm.

Definition.

For $0 < \beta < 1$ we say that a language L is in the class robust-$\mathrm{BP}_\beta\mathrm{P}$ if there is an $\varepsilon > 0$ such that for any positive probability measure ν satisfying

$$\sup_{w \in \{0,1\}^*, b \in \{0,1\}} |\nu(wb|w) - \mu^\beta(wb|w)| < \varepsilon$$

we have $L \in \mathrm{BP}_\nu\mathrm{P}$.

It follows from Vazirani and Vazirani [76] that BPP $\subseteq$ robust-$\mathrm{BP}_\beta\mathrm{P}$ so the class is not empty. It follows from Theorem 3.1.11 that robust-$\mathrm{BP}_\beta\mathrm{P}$ $\subseteq$ BPP and thus the following theorem holds.

Theorem 3.2.1.

For all $0 < \beta < 1$ we have robust-$\mathrm{BP}_\beta\mathrm{P} = \mathrm{BPP}$.

Proof.

The inclusion BPP $\subseteq$ robust-$\mathrm{BP}_\beta\mathrm{P}$ follows from [76].

To see the reverse inclusion, let $0 < \beta < 1$ be given and let $L \in$ robust-$\mathrm{BP}_\beta\mathrm{P}$. Let ε be as given in the definition of robust-$\mathrm{BP}_\beta\mathrm{P}$. Choose $0 < \gamma < 1$ satisfying $\beta - \varepsilon < \gamma < \beta + \varepsilon$ and $\gamma \in \mathrm{BPP}_{\mathrm{CF}}$. This is possible since the BPP-computable real numbers are dense in $(0,1)$. Then $|\mu^\beta(wb|w) - \mu^\gamma(wb|w)| < \varepsilon$ for every string w and bit b, thus $L \in \mathrm{BP}_\gamma\mathrm{P}$. Since γ is BPP-computable by Theorem 3.1.11 $\mathrm{BP}_\gamma\mathrm{P} = \mathrm{BPP}$. Thus we have $L \in \mathrm{BPP}$.
$\square$

This theorem tells us that to each BPP algorithm there corresponds a *robust* algorithm that is insensitive to variations in the witness distribution. This in turn means that BPP can fruitfully be thought of as the class of 'feasibly' computable languages.

CHAPTER 4 EQUIVALENCE OF MEASURES OF COMPLEXITY CLASSES

In the previous chapter we worked with probability measures based on a single biased coin. In this chapter we will allow each string to be included or excluded from a language based on the flip of one of an infinite sequence of biased coins. The ith string s_i will be in the language with probability β_i. Given $\vec{\alpha}$ and $\vec{\beta}$, two sequences of biases that are p-computable and strongly positive (terms to be defined below), and a class of languages $\mathcal{C}$ closed under a specific type of truth-table reducibility (again to be defined below), we show that $\mu^{\vec{\alpha}}(\mathcal{C}) = 0 \iff \mu^{\vec{\beta}}(\mathcal{C}) = 0$. Thus any two strongly positive, p-computable bias sequences generate measures that agree on which sets having the specified closure property are measure 0.

4.1 Resource-Bounded ν-Measure

We begin with a review of the well-known notion of a martingale over a probability measure ν. Computable martingales were used by Schnorr [61, 62, 63, 64] in his investigations of randomness, and have more recently been used by Lutz [43] in the development of resource-bounded measure.

Definition. Let ν be a probability measure on $\{0,1\}^\infty$. Then a *ν-martingale* is a function $d : \{0,1\}^* \longrightarrow [0,\infty)$ such that, for all $w \in \{0,1\}^*$,

$$d(w)\nu(w) = d(w0)\nu(w0) + d(w1)\nu(w1). \tag{4.1}$$

If $\vec{\beta}$ is a sequence of biases, then a $\mu^{\vec{\beta}}$-martingale is simply called a *$\vec{\beta}$-martingale*. A μ-martingale is even more simply called a *martingale*. (That is, when the probability measure is not specified, it is assumed to be the uniform probability measure μ.)

Intuitively, a ν-martingale d is a "strategy for betting" on the successive bits of (the characteristic sequence of) a language $A \in \{0,1\}^\infty$. The real number $\nu(\lambda)$ is regarded as the amount of money that the strategy starts with. The real number $\nu(w)$ is the amount of money that the strategy has after betting on a prefix w of χ_A. The identity (4.1) ensures that the betting is "fair" in the sense that, if A is chosen according to the probability measure ν, then the expected amount of money is constant as the betting proceeds. (See [61, 62, 63, 64, 74, 43, 45, 44] for further discussion.) Of course, the "objective" of a strategy is to win a lot of money.

Definition. A ν-martingale d *succeeds* on a language $A \in \{0,1\}^\infty$ if

$$\limsup_{n \longrightarrow \infty} d(\chi_A[0..n-1]) = \infty.$$

The *success set* of a ν-martingale d is the set

$$S^\infty[d] = \{A \in \{0,1\}^\infty \mid d \text{ succeeds on } A\}.$$

We are especially interested in martingales that are computable within some resource bound. (Recall that the p-computability and p_2-computability of real valued functions were defined in section 2.1.)

Definition. Let ν be a probability measure on $\{0,1\}^\infty$.

1. A *p-ν-martingale* is a ν-martingale that is p-computable.

2. A *p_2-ν-martingale* is a ν-martingale that is p_2-computable.

A p-$\mu^{\vec{\beta}}$-martingale is called a p-$\vec{\beta}$-martingale, a p-μ-martingale is called a p-martingale, and similarly for p_2.

We now come to the fundamental ideas of resource-bounded ν-measure.

Definition. Let ν be a probability measure on $\{0,1\}^\infty$, and let $X \subseteq bininf$.

1. X has *p-ν-measure 0*, and we write $\nu_{\mathrm{p}}(X) = 0$, if there is a p-ν-martingale d such that $X \subseteq S^\infty[d]$.

2. X has *p-ν-measure 1*, and we write $\nu_{\mathrm{p}}(X) = 1$, if $\nu_{\mathrm{p}}(X^c) = 0$, where $X^c = \{0,1\}^\infty - X$.

The conditions $\nu_{\mathrm{p}_2}(X) = 0$ and $\nu_{\mathrm{p}_2}(X) = 1$ are defined analogously.

Definition. Let ν be a probability measure on $\{0,1\}^\infty$, and let $X \subseteq \{0,1\}^\infty$.

1. X has *ν-measure 0 in* E, and we write $\nu(X|\mathrm{E}) = 0$, if $\nu_{\mathrm{p}}(X \cap E) = 0$.

2. X has *ν-measure 1 in* E, and we write $\nu(X|\mathrm{E}) = 1$, if $\nu(X^c|\mathrm{E}) = 0$.

3. X has *ν-measure 0 in* E_2, and we write $\nu(X|\mathrm{E}_2) = 0$, if $\nu_{\mathrm{p}_2}(X \cap \mathrm{E}_2) = 0$.

4. X has *ν-measure 1 in* E_2, and we write $\nu(X|\mathrm{E}_2) = 1$, if $\nu(X^c|\mathrm{E}_2) = 0$.

Just as in the uniform case [43], the resource bounds p and p_2 of the above definitions are only two possible values of a very general parameter. Other choices of this parameter yield classical ν-measure [33], constructive ν-measure (as used in algorithmic information theory [83, 74]), ν-measure in the set REC, consisting of all decidable languages, ν-measure in ESPACE, etc.

The rest of this section is devoted to a very brief presentation of some of the fundamental theorems of resource-bounded ν-measure. One of the main objectives of these results is to justify the intuition that *a set with ν-measure 0 in* E *contains only a "negligibly small" part of* E (with respect to ν). For the purpose of this chapter, it suffices to present these results for p-ν-measure and ν-measure in E. We note, however, that all these results hold *a fortiori* for p_2-ν-measure, rec-ν-measure, classical ν-measure, ν-measure in E_2, ν-measure in ESPACE, etc.

We first note that ν-measure 0 sets exhibit the set-theoretic behavior of small sets.

Definition. Let $X, X_0, X_1, X_2, \ldots \subseteq \{0,1\}^\infty$.

1. X is a p-*union of the* p-ν-*measure 0 sets* $X_0, X_1, X_2, \ldots$ if $X = \cup_{k=0}^{\infty} X_k$ and there is a sequence $d_0, d_1, d_2, \ldots$ of ν-martingales with the following two properties.
   (i) For each $k \in \mathbf{N}$, $X_k \subseteq S^\infty[d_k]$.
   (ii) The function $(k, w) \mapsto d_k(w)$ is p-computable.
2. X is a p-*union of the sets* $X_0, X_1, X_2, \ldots$ *of* ν-*measure 0 in* E if $X = \cap_{k=0}^{\infty} X_k$ and there is a sequence $d_0, d_1, d_2, \ldots$ of ν-martingales with the following two properties.
   (i) For each $k \in \mathbf{N}$, $X_k \cap \mathrm{E} \subseteq S^\infty[d_k]$.
   (ii) The function $(k, w) \mapsto d_k(w)$ is p-computable.

Lemma 4.1.1. Let ν be a probability measure on $\{0,1\}^\infty$, and let $\mathcal{I}$ be either the collection of all p-ν-measure 0 subsets of $\{0,1\}^\infty$, or the collection of all subsets of $\{0,1\}^\infty$that have ν-measure 0 in E. Then $\mathcal{I}$ has the following three closure properties.

1. If $X \subseteq Y \in \mathcal{I}$, then $X \in \mathcal{I}$.
2. If X is a finite union of elements of $\mathcal{I}$, then $X \in \mathcal{I}$
3. If X is a p-union of elements of $\mathcal{I}$, then $X \in \mathcal{I}$.

Proof (sketch). Assume that X is a p-union of the p-ν-measure 0 sets $X_0, X_1, X_2, \ldots$, and let $d_0, d_1, d_2, \ldots$ be as in the definition of this condition. Without loss of generality, assume that $d_k(\lambda) > 0$ for each $k \in \mathbf{N}$. It suffices to show that $\nu_{\mathrm{p}}(X) = 0$. (The remaining parts of the lemma are obvious or follow directly from this.) Define

$$d : \{0,1\}^* \longrightarrow [0, \infty)$$

$$d_k(w) = \sum_{k=0}^{\infty} \frac{d_k(w)}{2^k \cdot d_k(\lambda)}.$$

Its is easily checked that d is a p-ν-martingale and that $X \subseteq S^{\infty}[d]$, so $\nu_{\mathrm{p}}(X) = 0$. □

We next note that, if ν is strongly positive and p-computable, then every singleton subset of E has p-ν-measure 0.

Lemma 4.1.2. If ν is a strongly positive, p-computable probability measure on $\{0,1\}^{\infty}$, then for every $A \in \mathrm{E}$,

$$\nu_{\mathrm{p}}(\{A\}) = \nu(\{A\}|\mathrm{E}) = 0.$$

Proof (sketch). Assume the hypothesis, and fix $\delta > 0$ such that, for all $w \in \{0,1\}^*$ and $b \in \{0,1\}$, $\nu(wb|w) \geq \delta$. Define

$$d : \{0,1\}^* \longrightarrow [0,\infty)$$

$$d(\lambda) = 1$$

$$d(wb) = \frac{d(w)}{\nu(wb|w)} \cdot [\![s_{|w|} \in A]\!].$$

It is easily checked that d is a p-ν-martingale and that, for all $n \in \mathbf{N}$, $d(\chi_A[0..n-1]) \geq (1-\delta)^{-n}$, whence $A \in S^{\infty}[d]$. □.

Note that, for $A \in \mathrm{E}$, the "point-mass" probability measure

$$\pi_A : \{0,1\}^* \longrightarrow [0,1]$$

$$\pi_A(w) = \begin{cases} 1 & \text{if } w \sqsubseteq \chi_A \\ 0 & \text{if } w \not\sqsubseteq \chi_A \end{cases}$$

is p-computable, and $\{A\}$ does *not* have p-π_A-measure 0. Thus, the strong positivity hypothesis cannot be removed from Lemma 4.1.2.

We now come to the most crucial issue in the development of resource-bounded measure. If a set X has ν-measure 0 in E, then we want to say that X contains only a "negligible small" part of E. In particular, then, it is critical that E itself not have ν-measure 0 in E. The following theorem establishes this and more.

Theorem 4.1.3. Let ν be a probability measure on $\{0,1\}^\infty$, and let $w \in \{0,1\}^*$. If $\nu(w) > 0$, then $\mathbf{C}_w$ does not have ν-measure 0 in E.

Proof (sketch). Assume the hypothesis, and let d be a p-ν-martingale. It suffices to show that $\mathbf{C}_w \cap \mathrm{E} \not\subseteq S^\infty[d]$.

Since d is p-computable, there is a function $\hat{d} : \mathbf{N} \times \{0,1\}^* \longrightarrow \mathbf{Q} \cap [0,\infty)$ with the following two properties.

(i) For all $r \in \mathbf{N}$ and $w \in \{0,1\}^*$, $|\hat{d}(r,w) - d(w)| \leq 2^{-r}$.

(ii) There is an algorithm that computes $\hat{d}(r,w)$ in time polynomial in $r + |w|$.

Define a language A recursively as follows. First, for $0 \leq i < |w|$, $[\![s_i \in A]\!] = w[i]$. Next assume that the string $x_i = \chi_A[0..i-1]$ has been defined, where $i \geq |w|$. Then

$$[\![s_i \in A]\!] = [\![\hat{d}(i+1, x_i 1) \leq \hat{d}(i+1, x_i 0)]\!].$$

With the language A so defined, it is easy to check that $A \in \mathbf{C}_w \cap \mathrm{E}$. It is also routine to check that, for all $i \geq |w|$,

$$\begin{aligned} d(x_{i+1}) &\leq \hat{d}(i+1, x_{i+1}) + 2^{-(i+1)} \\ &= \min\left\{\hat{d}(i+1, x_i 0), \hat{d}(i+1, x_i 1)\right\} + 2^{-(i+1)} \\ &\leq \min\{d(x_i 0), d(x_i 1)\} + 2^{-i} \\ &\leq d(x_i) + 2^{-i}. \end{aligned}$$

It follows inductively that, for all $n \geq |w|$,

$$\begin{aligned} d(x_n) &\leq d(w) + \sum_{i=|w|}^{n-1} 2^{-i} \\ &< d(w) + \sum_{i=|w|}^{\infty} 2^{-i} = d(w) + 2^{1-|w|}. \end{aligned}$$

This implies that

$$\limsup_{n \longrightarrow \infty} d(\chi_A[0..n-1]) \leq d(w) + 2^{1-|w|} < \infty,$$

whence $A \notin S^\infty[d]$. □

As in the case of the uniform probability measure [43], more quantitative results on resource-bounded ν-measure can be obtained by considering the *unitary success set*

$$S^1[d] = \bigcup_{\substack{w \\ d(w) \geq 1}} \mathbf{C}_w$$

and the *initial value* $d(\lambda)$ of a p-ν-martingale d. For example, generalizing the arguments in [43] in a straightforward manner, this approach yields a Measure Conservation Theorem for ν-measure (a quantitative extension of Theorem 4.1.3) and a uniform, resource-bounded extension of the classical first Borel-Cantelli lemma. As these results are not used in the present dissertation, we refrain from elaborating here.

4.2 Summable Equivalence

If two probability measures on $\{0,1\}^\infty$ are sufficiently "close" to one another, then the Summable Equivalence Theorem says that the two probability measures are in absolute agreement as to which sets of languages have p-measure 0 and which do not. In this section, we define this notion of "close" and prove this result.

Definition. Let ν be a positive probability measure on $\{0,1\}^\infty$, let $A \subseteq \{0,1\}^*$, and let $i \in \mathbf{N}$. Then the i^{th} *conditional ν-probability along A* is

$$\nu_A(i+1|i) = \nu(\chi_A[0..i] \mid \chi_A[0..i-1]).$$

Definition. Two positive probability measures ν and ν' on $\{0,1\}^\infty$ are *summably equivalent*, and we write $\nu \approx \nu'$, if for every $A \subseteq \{0,1\}^*$,

$$\sum_{i=0}^{\infty} |\nu_A(i+1|i) - \nu'_A(i+1|i)| < \infty.$$

It is clear that summable equivalence is an equivalence relation on the collection of all positive probability measures on $\{0,1\}^\infty$. The following fact is also easily verified.

Lemma 4.2.1. Let ν and ν' be positive probability measures on $\{0,1\}^\infty$. If $\nu \approx \nu'$, then ν is strongly positive if and only if ν' is strongly positive.

The following definition gives the most obvious way to transform a martingale for one probability measure into a martingale for another.

Definition. Let ν and ν' be probability measures on $\{0,1\}^\infty$ with ν' positive, and let d be a ν-martingale. Then the *canonical adjustment* of d to ν' is the ν'-martingale d' defined by

$$d'(w) = \frac{\nu(w)}{\nu'(w)} d(w)$$

for all $w \in \{0,1\}^*$.

It is trivial to check that the above function d' is indeed a ν'-martingale. The following lemma shows that, for strongly positive probability measures, summable equivalence is a sufficient condition for d' to succeed whenever d succeeds.

Lemma 4.2.2. Let ν and ν' be strongly positive probability measures on $\{0,1\}^\infty$, let d be a ν-martingale, and let d' be the canonical adjustment of d to ν'. If $\nu \approx \nu'$, then $S^\infty[d] \subseteq S^\infty[d']$.

Proof. Assume the hypothesis, and let $A \in S^\infty[d]$. For each $i \in \mathbf{N}$, let

$$\nu_i = \nu_A(i+1|i), \quad \nu'_i = \nu'_A(i+1|i), \quad \tau_i = \nu_i - \nu'_i.$$

The hypothesis $\nu \approx \nu'$ says that $\sum_{i=0}^\infty |\tau_i| < \infty$. In particular, this implies that $\tau_i \longrightarrow 0$ as $i \longrightarrow \infty$, so we have the Taylor approximation

$$\ln \frac{\nu_i}{\nu'_i} = \ln(1 + \frac{\tau_i}{\nu'_i}) = \frac{\tau_i}{\nu'_i} + \mathrm{o}(\frac{\tau_i}{\nu'_i})$$

as $i \longrightarrow \infty$. Thus $|\ln \frac{\nu_i}{\nu'_i}|$ is asymptotically equivalent to $\frac{|\tau_i|}{\nu'_i}$ as $i \longrightarrow \infty$. Since ν' is strongly positive, it follows that $\sum_{i=0}^\infty |\ln \frac{\nu_i}{\nu'_i}| < \infty$. Thus, if we let $w_k = \chi_A[0..k-1]$, then there is a positive constant c such that, for all $k \in \mathbf{N}$,

$$c \geq \sum_{i=0}^{k-1}(-\ln \frac{\nu_i}{\nu_i'}) = -\ln \prod_{i=0}^{k-1} \frac{\nu_i}{\nu_i'} = -\ln \frac{\nu(w_k)}{\nu'(w_k)},$$

whence

$$d'(w_k) = \frac{\nu(w_k)}{\nu'(w_k)} d(w_k) \geq e^{-c} d(w_k).$$

Since $A \in S^\infty[d]$, we thus have

$$\limsup_{k \longrightarrow \infty} d'(w_k) \geq \limsup_{k \longrightarrow \infty} e^{-c} d(w_k) = \infty,$$

so $A \in S^\infty[d']$. □

The following useful result is now easily established.

Theorem 4.2.3 (Summable Equivalence Theorem). [1] If ν and ν' are strongly positive, p-computable probability measures on $\{0,1\}^\infty$ such that $\nu \approx \nu'$, then for every set $X \subseteq \{0,1\}^\infty$,

$$\nu_{\mathrm{p}}(X) = 0 \Longleftrightarrow \nu'_{\mathrm{p}}(X) = 0.$$

Proof. Assume the hypothesis, and assume that $\nu_{\mathrm{p}}(X) = 0$. By symmetry, it suffices to show that $\nu'_{\mathrm{p}}(X) = 0$. Since $\nu_{\mathrm{p}}(X) = 0$, there is a p-computable ν-martingale d such that $X \subseteq S^\infty[d]$. Let d' be the canonical adjustment of d to ν'. Since $d, \nu,$ and ν' are all p-computable, it is easy to see that d' is p-computable. Since $\nu \approx \nu'$, Lemma 4.2.2 tells us that

$$X \subseteq S^\infty[d] \subseteq S^\infty[d'].$$

Thus $\nu'_{\mathrm{p}}(X) = 0$. □

[1] Kautz [39], answering a question in [20], has recently proven a stronger version of this theorem in which the hypothesis $\nu \approx \nu'$ is weakened to $\sum_{i=0}^{\infty} |\nu_A(i+1|i) - \nu'_A(i+1|i)|^2 < \infty$.

4.3 Exact Computation

It is sometimes useful or convenient to work with probability measures that are rational-valued and efficiently computable in an exact sense, with no approximation. This section presents two very easy results identifying situations in which such probability measures are available.

Definition. A probability measure ν on $\{0,1\}^\infty$is *exactly* p-*computable* if $\nu : \{0,1\}^* \longrightarrow \mathbf{Q} \cap [0,1]$ and there is an algorithm that computes $\nu(w)$ in time polynomial in $|w|$.

Lemma 4.3.1. For every strongly positive, p-computable probability measure ν on $\{0,1\}^\infty$, there is an exactly p-computable probability measure ν' on $\{0,1\}^\infty$such that $\nu \approx \nu'$.

Proof. Let ν be a p-computable probability measure on $\{0,1\}^\infty$, and fix a function $\hat{\nu} : \mathbf{N} \times \{0,1\}^* \longrightarrow \mathbf{Q} \cap [0,1]$ that testifies to the p-computability of ν. Since ν is strongly positive, there is a constant $c \in \mathbf{N}$ such that, for all $w \in \{0,1\}^*$, $2^{-c|w|} \leq \nu(w) \leq 1 - 2^{-c|w|}$. Fix such a c and, for all $w \in \{0,1\}^*$, define

$$\nu'(w0|w) = \min\left\{1, \frac{\hat{\nu}((2c+1)|w|+3, w0)}{\hat{\nu}((2c+1)|w|+3, w)}\right\},$$

$$\nu'(w1|w) = 1 - \nu'(w0|w),$$

$$\nu'(w) = \prod_{i=0}^{|w|-1} \nu'(w[0..i]|w[0..i-1]).$$

It is clear that ν' is an exactly p-computable probability measure on $\{0,1\}^\infty$.

Now let $w \in \{0,1\}^*$ and $b \in \{0,1\}$. For convenience, let

$$\begin{aligned} \delta &= 2^{-(1+c|w|)}, \\ \epsilon &= 2^{-(2c+1)|w|-3}, \\ a_1 &= \nu(wb), \\ a_2 &= \nu(w). \end{aligned}$$

Note that

$$\hat{\nu}((2c+1)|w|+3, w) \geq \nu(w) - \epsilon > \nu(w) - \delta \geq \delta.$$

It is clear by inspection that $\nu'(wb|w)$ can be written in the form

$$\nu'(wb|w) = \frac{a_1'}{a_2'},$$

where

$$|a_1' - a_1| \leq \epsilon \ \text{ and } \ |a_2' - a_2| \leq \epsilon.$$

We thus have

$$\begin{aligned} |a_1' a_2 - a_1 a_2'| &\leq |a_1' a_2 - a_1 a_2| + |a_1 a_2 - a_1 a_2'| \\ &\leq |a_1' - a_1| + |a_2' - a_2| \\ &\leq 2\epsilon, \end{aligned}$$

whence

$$\begin{aligned} |\nu'(wb|w) - \nu(wb|w)| &= \left| \frac{a_1'}{a_2'} - \frac{a_1}{a_2} \right| \\ &= \frac{|a_1' a_2 - a_1 a_2'|}{a_2 a_2'} \\ &\leq 2\epsilon\delta^{-2} \\ &= 2^{-|w|}. \end{aligned}$$

For all $A \subseteq \{0,1\}^*$, then, we have

$$\sum_{i=0}^{\infty} |\nu_A(i+1|i) - \nu_A'(i+1|i)| \ \leq \ \sum_{i=0}^{\infty} 2^{-i} \ = \ 2,$$

so $\nu \approx \nu'$. □

For some purposes (including those of this chapter), the requirement of p-computability is too weak, because it allows $\nu(w)$ to be computed (or approximated) in time polynomial in $|w|$, which is exponential in the length of the last string decided by w

when we regard w as a prefix of a language A. In such situations, the following sort of requirement is often more useful. (We only give the definitions for sequences of biases, i.e., coin-toss probability measures, because this suffices for our purposes in this dissertation. It is clearly a routine matter to generalize further.)

Definition.

1. A P-*sequence of biases* is a sequence $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \ldots)$ of biases $\beta_i \in [0,1]$ for which there is a function

$$\hat{\beta} : \mathbf{N} \times \mathbf{N} \longrightarrow \mathbf{Q} \cap [0,1]$$

with the following two properties.

(i) For all $i, r \in \mathbf{N}$, $|\hat{\beta}(i,r) - \beta_i| \leq 2^{-r}$.

(ii) There is an algorithm that, for all $i, r \in \mathbf{N}$, computes $\hat{\beta}(i,r)$ in time polynomial in $|s_i| + r$ (i.e., in time polynomial in $\log(i+1) + r$).

2. A P-*exact sequence of biases* is a sequence $\vec{\beta} = (\beta_0, \beta_1, \beta_2, \ldots)$ of (rational) biases $\beta_i \in \mathbf{Q} \cap [0,1]$ such that the function $i \longmapsto \beta_i$ is computable in time polynomial in $|s_i|$.

Definition. If $\vec{\alpha}$ and $\vec{\beta}$ are sequences of biases, then $\vec{\alpha}$ and $\vec{\beta}$ are *summably equivalent*, and we write $\vec{\alpha} \approx \vec{\beta}$, if $\sum_{i=0}^{\infty} |\alpha_i - \beta_i| < \infty$.

It is clear that $\vec{\alpha} \approx \vec{\beta}$ if and only if $\mu^{\vec{\alpha}} \approx \mu^{\vec{\beta}}$.

Lemma 4.3.2. For every P-sequence of biases $\vec{\beta}$, there is a P-exact sequence of biases $\vec{\beta}'$ such that $\vec{\beta} \approx \vec{\beta}'$.

Proof. Let $\vec{\beta}$ be a strongly positive P-sequence of biases, and let $\hat{\beta} : \mathbf{N} \times \mathbf{N} \longrightarrow \mathbf{Q} \cap [0,1]$ be a function that testifies to this fact. For each $i \in \mathbf{N}$, let

$$\beta_i' = \hat{\beta}(i, 2|s_i|),$$

and let $\vec{\beta}' = (\beta'_0, \beta'_1, \beta'_2, \cdots)$. Then $\vec{\beta}'$ is a P-exact sequence of biases, and

$$\begin{aligned}\sum_{i=0}^{\infty} |\beta_i - \beta'_i| &\le \sum_{i=0}^{\infty} 2^{-2|s_i|} \\ &\le \sum_{i=0}^{\infty} 2^{-2\log(i+1)} \\ &= \sum_{i=0}^{\infty} \frac{1}{(i+1)^2} < \infty,\end{aligned}$$

so $\vec{\beta} \approx \vec{\beta}'$. □

4.4 Martingale Dilation

In this section we show that certain truth-table reductions can be used to *dilate* martingales for one probability measure into martingales for another, perhaps dissimilar, probability measure on $\{0,1\}^\infty$. We first present some terminology and notation on truth-table reductions. (Most of this notation is standard [58], but some is specialized to our purposes.)

A *truth-table reduction* (briefly, a $\le_{\rm tt}$-reduction) is an ordered pair (f, g) of total recursive functions such that for each $x \in \{0,1\}^*$, there exists $n(x) \in \mathbf{Z}^+$ such that the following two conditions hold.

(i) $f(x)$ is (the standard encoding of) an $n(x)$-tuple $(f_1(x), \ldots, f_{n(x)}(x))$ of strings $f_i(x) \in \{0,1\}^*$, which are called the *queries* of the reduction (f, g) on input x. We use the notation $Q_{(f,g)}(x) = \{f_1(x), \ldots, f_{n(x)}(x)\}$ for the set of such queries.

(ii) $g(x)$ is (the standard encoding of) an $n(x)$-input, 1-output Boolean circuit, called the *truth table* of the reduction (f, g) on input x. We identify $g(x)$ with the Boolean function computed by this circuit, i.e.,

$$g(x) : \{0,1\}^{n(x)} \longrightarrow \{0,1\}.$$

A truth-table reduction (f, g) *induces* the function

$$F_{(f,g)} : \{0,1\}^\infty \longrightarrow \{0,1\}^\infty$$

$$F_{(f,g)}(A) = \left\{x \in \{0,1\}^* \mid g(x)\left([\![f_1(x) \in A]\!] \cdots [\![f_{n(x)}(x) \in A]\!]\right) = 1\right\}.$$

If A and B are languages and (f, g) is a $\leq_{\mathrm{tt}}$-reduction, then (f, g) *reduces* B *to* A, and we write

$$B \leq_{\mathrm{tt}} A \text{ via } (f, g),$$

if $B = F_{(f,g)}(A)$. More generally, if A and B are languages, then B is *truth-table reducible* (briefly, $\leq_{\mathrm{tt}}$-reducible) to A, and we write $B \leq_{\mathrm{tt}} A$, if there exists a $\leq_{\mathrm{tt}}$-reduction (f, g) such that $B \leq_{\mathrm{tt}} A$ via (f, g).

If (f, g) is a $\leq_{\mathrm{tt}}$-reduction, then the function $F_{(f,g)} : \{0,1\}^\infty \longrightarrow \{0,1\}^\infty$ defined above induces a corresponding function

$$F_{(f,g)} : \{0,1\}^* \longrightarrow \{0,1\}^* \cup \{0,1\}^\infty$$

defined as follows. (It is standard practice to use the same notation for these two functions, and no confusion will result from this practice here.) Intuitively, if $A \in \{0,1\}^\infty$ and $w \sqsubseteq A$, then $F_{(f,g)}(w)$ is the largest prefix of $F_{(f,g)}(A)$ such that w answers all queries in this prefix. Formally, let $w \in \{0,1\}^*$, and let

$$A_w = \{s_i \mid 0 \leq i < |w| \text{ and } w[i] = 1\}.$$

If $Q_{(f,g)}(x) \subseteq \{s_0, \ldots s_{|w|-1}\}$ for all $x \in \{0,1\}^*$, then

$$F_{(f,g)}(w) = F_{(f,g)}(A_w).$$

Otherwise,

$$F_{(f,g)}(w) = \chi_{F_{(f,g)}(A_w)}[0..m-1],$$

where m is the greatest nonnegative integer such that

$$\bigcup_{i=0}^{m-1} Q_{(f,g)}(s_i) \subseteq \left\{s_0, \ldots, s_{|w|-1}\right\}$$

Now let (f,g) be a $\leq_{\mathrm{tt}}$-reduction, and let $z \in \{0,1\}^*$. Then the *inverse image* of the cylinder $\mathbf{C}_z$ under the reduction (f,g) is

$$\begin{aligned} F^{-1}_{(f,g)}(\mathbf{C}_z) &= \left\{A \in \{0,1\}^\infty \mid F_{(f,g)}(A) \in \mathbf{C}_z\right\} \\ &= \left\{A \in \{0,1\}^\infty \mid z \sqsubseteq F_{(f,g)}(A)\right\}. \end{aligned}$$

We can write this set in the form

$$F^{-1}_{(f,g)}(\mathbf{C}_z) = \bigcup_{w \in I} \mathbf{C}_w,$$

where I is the set of all strings $w \in \{0,1\}^*$ with the following properties.

(i) $z \sqsubseteq F_{(f,g)}(w)$.

(ii) If w' is a proper prefix of w, then $z \not\sqsubseteq F_{(f,g)}(w')$.

Moreover, the cylinders $\mathbf{C}_w$ in this union are disjoint, so if ν is a probability measure on $\{0,1\}^\infty$, then

$$\nu(F^{-1}_{(f,g)}(\mathbf{C}_z)) = \sum_{w \in I} \nu(w).$$

The following well-known fact is easily verified.

Lemma 4.4.1. If ν is a probability measure on $\{0,1\}^\infty$ and (f,g) is a $\leq_{\mathrm{tt}}$-reduction, then the function

$$\nu^{(f,g)} : \{0,1\}^* \longrightarrow [0,1]$$

$$\nu^{(f,g)}(z) = \nu(F^{-1}_{(f,g)}(\mathbf{C}_z))$$

is also a probability measure on $\{0,1\}^\infty$.

The probability measure $\nu^{(f,g)}$ of Lemma 4.4.1 is called the *probability measure induced by ν and (f,g)*.

In this chapter, we only use the following special type of $\leq_{\mathrm{tt}}$-reduction.

Definition. A $\leq_{\mathrm{tt}}$-reduction (f,g) is *orderly* if, for all $x,y,u,v \in \{0,1\}^*$, if $x < y$, $u \in Q_{(f,g)}(x)$, and $v \in Q_{(f,g)}(y)$, then $u < v$. That is, if x precedes y (in the standard ordering of $\{0,1\}^*$), then every query of (f,g) on input x precedes every query of (f,g) on input y.

The following is an obvious property of orderly $\leq_{\mathrm{tt}}$-reductions.

Lemma 4.4.2. If ν is a coin-toss probability measure on $\{0,1\}^\infty$ and (f,g) is an orderly $\leq_{\mathrm{tt}}$-reduction, then $\nu^{(f,g)}$ is also a coin-toss probability measure on $\{0,1\}^\infty$.

Note that, if (f,g) is an orderly $\leq_{\mathrm{tt}}$-reduction, then $F_{(f,g)}(w) \in \{0,1\}^*$ for all $w \in \{0,1\}^*$. Note also that the length of $F_{(f,g)}(w)$ depends only upon the length of w (i.e., $|w| = |w'|$ implies that $|F_{(f,g)}(w)| = |F_{(f,g)}(w')|$). Finally, note that for each $m \in \mathbf{N}$ there exists $l \in \mathbf{N}$ such that $|F_{(f,g)}(0^l)| = m$.

Definition. Let (f,g) be an orderly $\leq_{\mathrm{tt}}$-reduction.

1. An (f,g)-*step* is a positive integer l such that $F_{(f,g)}(0^{l-1}) \neq F_{(f,g)}(0^l)$.
2. For $k \in \mathbf{N}$, we let $step(k)$ be the least (f,g)-step l such that $l \geq k$.

The following construction is crucial to the proof of our main theorem.

Definition. Let ν be a positive probability measure on $\{0,1\}^\infty$, let (f,g) be an orderly $\leq_{\mathrm{tt}}$-reduction, and let d be a $\nu^{(f,g)}$-martingale. Then the (f,g)-*dilation* of d is the function

$$(f,g)\hat{\ }d : \{0,1\}^* \longrightarrow [0,\infty)$$

$$(f,g)\hat{\ }d(w) = \sum_{u \in \{0,1\}^{l-k}} d(F_{(f,g)}(wu))\nu(wu|w),$$

where $k = |w|$ and $l = step(k)$.

In other words, $(f,g)\hat{\ }d(w)$ is the conditional ν-expected value of $d(F_{(f,g)}(w'))$, given that $w \sqsubseteq w'$ and $|w'| = step(|w|)$. We do not include the probability measure ν in the notation $(f,g)\hat{\ }d$ because ν (being positive) is implicit in d.

Intuitively, the function $(f,g)\hat{\ }d$ is a strategy for betting on a language A, assuming that d itself is a strategy for betting on the language $F_{(f,g)}(A)$. The following theorem makes this intuition precise.

Theorem 4.4.3 (Martingale Dilation Theorem). Assume that ν is a positive coin-toss probability measure on $\{0,1\}^\infty$, (f,g) is an orderly $\leq_{\mathrm{tt}}$-reduction, and d is a $\nu^{(f,g)}$-martingale. Then $(f,g)\hat{\ }d$ is a ν-martingale. Moreover, for every language $A \subseteq \{0,1\}^*$, if d succeeds on $F_{(f,g)}(A)$, then $(f,g)\hat{\ }d$ succeeds on A.

A very special case of the above result (for strictly increasing $\leq_{\mathrm{m}}^{\mathrm{P}}$-reductions under the uniform probability measure) was developed by Ambos-Spies, Terwijn, and Zheng [3], and made explicit by Juedes and Lutz [37]. Our use of martingale dilation in the present chapter is very different from the simple padding arguments of [3, 37].

The following two technical lemmas are used in the proof of Theorem 4.4.3.

Lemma 4.4.4. Assume that ν is a positive coin-toss probability measure on $\{0,1\}^\infty$ and (f,g) is an orderly $\leq_{\mathrm{tt}}$-reduction. Let $F = F_{(f,g)}$, let $w \in \{0,1\}^*$, and assume that $k = |w|$ is an (f,g)-step. Let $l = step(k+1)$. Then, for $b \in \{0,1\}$,

$$\nu^{(f,g)}(F(w)b|F(w)) = \sum_{\substack{u \in \{0,1\}^{l-k} \\ F(wu) = F(w)b}} \nu(wu|w).$$

Proof. Assume the hypothesis. Then

$$\begin{aligned} \nu^{(f,g)}(F(w)b) &= \sum_{\substack{w' \in \{0,1\}^k \\ F(w') = F(w)}} \sum_{\substack{u \in \{0,1\}^{l-k} \\ F(w'u) = F(w')b}} \nu(w'u) \\ &= \sum_{\substack{w' \in \{0,1\}^k \\ F(w') = F(w)}} \nu(w') \sum_{\substack{u \in \{0,1\}^{l-k} \\ F(w'u) = F(w')b}} \nu(w'u|w'). \end{aligned}$$

Now, since ν is a coin-toss probability measure, we have $\nu(w'u|w') = \nu(wu|w)$ for each $w' \in \{0,1\}^k$ such that $F(w') = F(w)$. Also, since (f,g) is orderly, the conditions $F(w'u) = F(w')b$ and $F(wu) = F(w)b$ are equivalent for each $u \in \{0,1\}^{l-k}$. Hence,

$$\begin{aligned}
\nu^{(f,g)}(F(w)b) &= \sum_{\substack{w' \in \{0,1\}^k \\ F(w') = F(w)}} \nu(w') \sum_{\substack{u \in \{0,1\}^{l-k} \\ F(wu) = F(w)b}} \nu(wu|w) \\
&= \nu^{(f,g)}(F(w)) \sum_{\substack{u \in \{0,1\}^{l-k} \\ F(wu) = F(w)b}} \nu(wu|w).
\end{aligned}$$

□

Lemma 4.4.5. Assume that ν is a positive coin-toss probability measure on $\{0,1\}^\infty$ and (f,g) is an orderly $\leq_{\mathrm{tt}}$-reduction. Let $F = F_{(f,g)}$, and assume that d is a $\nu^{(f,g)}$-martingale. Let $w \in \{0,1\}^*$, assume that $k = |w|$ is an (f,g)-step, and let $l = step(k+1)$. Then

$$d(F(w)) = \sum_{u \in \{0,1\}^{l-k}} d(F(wu))\nu(wu|w).$$

Proof. Assume the hypothesis. Since d is a $\nu^{(f,g)}$-martingale and $\nu^{(f,g)}(F(w))$ is positive, we have

$$d(F(w)) = \sum_{b=0}^{1} d(F(w)b)\nu^{(f,g)}(F(w)b|F(w)).$$

It follows by Lemma 4.4.4 that

$$\begin{aligned}
d(F(w)) &= \sum_{b=0}^{1} d(F(w)b) \sum_{\substack{u \in \{0,1\}^{l-k} \\ F(wu) = F(w)b}} \nu(wu|w) \\
&= \sum_{b=0}^{1} \sum_{\substack{u \in \{0,1\}^{l-k} \\ F(wu) = F(w)b}} d(F(wu))\nu(wu|w) \\
&= \sum_{u \in \{0,1\}^{l-k}} d(F(wu))\nu(wu|w).
\end{aligned}$$

□

Proof of Theorem 4.4.3. Assume the hypothesis, and let $F = F_{(f,g)}$.

To see that $(f,g)\hat{}d$ is a ν-martingale, let $w \in \{0,1\}^*$, let $k = |w|$, and let $l = step(k+1)$. We have two cases.

Case I. $step(k) = l$. Then

$$\begin{aligned}
\sum_{b=0}^{1}(f,g)\hat{}d(wb)\nu(wb) &= \sum_{b=0}^{1}\sum_{u\in\{0,1\}^{l-k-1}} d(F(wbu))\nu(wbu|wb)\nu(wb) \\
&= \sum_{b=0}^{1}\sum_{u\in\{0,1\}^{l-k-1}} d(F(wbu))\nu(wbu) \\
&= \sum_{u\in\{0,1\}^{l-k}} d(F(wu))\nu(wu) \\
&= (f,g)\hat{}d(w)\nu(w).
\end{aligned}$$

Case II. $step(k) < l$. Then k is an (f,g)-step, so $(f,g)\hat{}d(w) = d(F(w))$, whence by Lemma 4.4.5

$$(f,g)\hat{}d(w)\nu(w) = \sum_{u\in\{0,1\}^{l-k}} d(F(wu))\nu(wu).$$

Calculating as in Case I, it follows that

$$(f,g)\hat{}d(w)\nu(w) = \sum_{b=0}^{1}(f,g)\hat{}d(wb)\nu(wb).$$

This completes the proof that $(f,g)\hat{}d$ is a ν-martingale.

To complete the proof, let $A \subseteq \{0,1\}^*$, and assume that d succeeds on $F(A)$. For each $n \in \mathbf{N}$, let $w_n = \chi_A[0..l_n-1]$, where l_n is the unique (f,g)-step such that $|F(0^{l_n})| = n$. Then, for all $n \in \mathbf{N}$,

$$(f,g)\hat{}d(w_n) = d(F(w_n)) = d(\chi_{F(A)}[0..n-1]),$$

so

$$\begin{aligned}
\limsup_{k\longrightarrow\infty}(f,g)\hat{}d(\chi_A[0..k-1]) &\geq \limsup_{n\longrightarrow\infty}(f,g)\hat{}d(w_n) \\
&= \limsup_{n\longrightarrow\infty} d(\chi_{F(A)}[0..n-1]) \\
&= \infty.
\end{aligned}$$

Thus $(f,g)\hat{}d$ succeeds on A. $\square$

4.5 Positive Bias Reduction

In this section, we define and analyze a positive truth-table reduction that encodes an efficient, approximate simulation of one sequence of biases by another.

Intuitively, if $\vec{\alpha}$ and $\vec{\beta}$ are strongly positive sequences of biases, then the positive reduction of $\vec{\alpha}$ to $\vec{\beta}$ is a $\leq_{\mathrm{tt}}$-reduction (f, g) that "tries to simulate" the sequence $\vec{\alpha}$ with the sequence $\vec{\beta}$ by causing $\mu^{\vec{\alpha}}$ to be the probability distribution induced by $\mu^{\vec{\beta}}$ and (f, g). In general, this objective will only be approximately achieved, in the sense that the probability distribution induced by $\mu^{\vec{\beta}}$ and (f, g) will actually be a probability distribution $\mu^{\vec{\alpha}'}$, where $\vec{\alpha}'$ is a sequence of biases such that $\vec{\alpha}' \approx \vec{\alpha}$.

The reduction (f, g) is constructed precisely as follows.

Construction 4.5.1 (Positive Bias Reduction). Let $\vec{\alpha}$ and $\vec{\beta}$ be strongly positive sequences of biases. Let

$$\delta = \inf\ \{\alpha_i, 1 - \alpha_i, \beta_i, 1 - \beta_i \mid i \in \mathbf{N}\},$$

$$c = \lceil \frac{-4 \log e}{\log(1 - \delta^2)} \rceil.$$

For each $x \in \{0,1\}^*$ and $0 \leq n < 2^{c|x|}$, let $q(x, n) = xy$, where y is the n^{th} element of $\{0,1\}^{c|x|}$, and let $j(x, n)$ be the index of the string $q(x, n)$, i.e., $s_j(x, n) = q(x, n)$. Then the *positive bias reduction of $\vec{\alpha}$ to $\vec{\beta}$* is the ordered pair (f, g) of functions defined by the procedure in Figure 4.1. (For convenience, the procedure defines additional parameters that are useful in the subsequent analysis.)

The following general remarks will be helpful in understanding Construction 4.5.1.

(a) The boldface variables $\mathbf{v_0}, \mathbf{v_1}, \ldots$ denote Boolean inputs to the Boolean function $g(x)$ being constructed. The Boolean function $g(x)$ is an OR of $k(x)$ Boolean functions $h(x, k)$, i.e.,

$$g(x) = \bigvee_{k=0}^{k(x)-1} h(x, k).$$

begin
input $x = s_i$;
$n := 0$;
$g(x, 0) := 0$; $\alpha'_i(0) = 0$;
$k := 0$;
while $\alpha'_i(k) < \alpha_i - (i+1)^{-2}$ **do**
    begin
    $h(x, k, 0) := 1$; $\gamma_{i,k}(0) := 1$;
    $l := 0$;
    while $\alpha'_i(k) + \gamma_{i,k}(l) - \alpha'_i(k) \cdot \gamma_{i,k}(l) > \alpha_i$ **do**
        begin
        $h(x, k, l+1) := h(x, k, l)$ AND v_n;
        $\gamma_{i,k}(l+1) := \gamma_{i,k}(l) \cdot \beta_{j(x,n)}$;
        $l := l + 1$;
        $n := n + 1$;
        end ;
    $l(x, k) := l$;
    $h(x, k) := h(x, k, l(x, k))$;
    $\gamma_{i,k} := \gamma_{i,k}(l(x, k))$;
    $g(x, k+1) := g(x, k)$ OR $h(x, k)$;
    $\alpha'_i(k+1) := \alpha'_i(k) + \gamma_{i,k} - \alpha'_i(k) \cdot \gamma_{i,k}$;
    $k := k + 1$
    end ;
$k(x) := k$;
$n(x) := n$;
$f(x) := (q(x, 0), \ldots, q(x, n(x) - 1))$;
$g(x) := g(x, n(x))$;
$\alpha'_i := \alpha_i(k(x))$
end .

Figure 4.1 Construction of positive bias reduction

The Boolean functions $g(x,0), g(x,1), \ldots$ are preliminary approximations of the Boolean function $g(x)$. In particular,

$$g(x,k) = \bigvee_{k=0}^{k-1} h(x,j)$$

for all $0 \leq k \leq k(x)$. Thus $g(x,0)$ is the constant-0 Boolean function.

(b) The Boolean function $h(x,k)$ is an AND of $l(x,k)$ consecutive input variables. The subscript n is incremented globally so that no input variable appears more than once in $g(x)$. Just as $g(x,k)$ is the k^{th} "partial OR" of g(x), $h(x,k,l)$ is the l^{th} "partial AND" of $h(x,k)$. Thus $h(x,k,0)$ is the constant-1 Boolean function.

(c) The input variables $\mathbf{v}_0$, $\mathbf{v}_1, \ldots$ of g correspond to the respective queries $q(x,0)$, $q(x,1), \ldots$ of f. If $A = F_{(f,g)}(B)$, then we have $[\![x \in A]\!] = g(x)(\mathbf{v}_0 \cdots \mathbf{v}_{n(x)-1})$, where each $\mathbf{v}_n = [\![q(x,n) \in B]\!]$. If B is chosen according to the sequence of biases $\vec{\beta}$, then $\beta_{j(x,n)}$ is the probability that $\mathbf{v}_n = 1$, $\gamma_{i,k}$ is the probability that $h(x,k) = 1$, and α_i' is the probability that $g(x) = 1$. The while-loops ensure that $\alpha_i - (i+1)^{-2} \leq \alpha_i' \leq \alpha_i$.

The following lemmas provide some quantitative analysis of the behavior of Construction 4.5.1.

Lemma 4.5.2. In Construction 4.5.1, for all $x \in \{0,1\}^*$ and $0 \leq k \leq k(x)$,

$$l(x,k) \leq 1 + \frac{c|x|}{2\log e}.$$

Proof. Fix such x and k, and let $l^* = l(x,k)$. If $l^* = 0$, the result is trivial, so assume that $l^* > 0$. Then, by the minimality of l^*,

$$\alpha_i'(k) + \gamma_{i,k}(l^* - 1) > \alpha_i,$$

so

$$\gamma_{i,k}(l^* - 1) > \alpha_i - \alpha_i'(k) > (i+1)^{-2},$$

so

$$(i+1)^{-2} < \gamma_{i,k}(l^* - 1) \leq (1-\delta)^{l^*-1}.$$

It follows that

$$-2\log(i+1) \leq (l^* - 1)\log(1-\delta),$$

whence

$$\begin{aligned} l^* &\leq 1 - \frac{2\log(i+1)}{\log(1-\delta)} \\ &\leq 1 - \frac{2|x|}{\log(1-\delta^2)} \\ &\leq 1 + \frac{c|x|}{1\log e}. \end{aligned}$$

□

Lemma 4.5.3. In the Construction 4.5.1, for all $x \in \{0,1\}^*$, and $0 \leq k \leq k(x) - 1$,

$$\alpha_i - \alpha_i'(k) \leq (1-\delta^2)^k.$$

Proof. Fix such x and k with $k < k(x) - 1$, and let $l^* = l(x,h)$. Then $\gamma_{i,k}(l^* - 1) > \alpha_i - \alpha_i'(k)$, so $\gamma_{i,k} \geq \delta \cdot \gamma_{i,k}(l^* - 1) > \delta \cdot (\alpha_i - \alpha_i'(k))$, whence

$$\begin{aligned} \frac{\alpha_i - \alpha_i'(k+1)}{\alpha_i - \alpha_i'(k)} &= \frac{\alpha_i - (\alpha_i'(k) + \gamma_{i,k} - \alpha_i'(k) \cdot \gamma_{i,k})}{\alpha_i - \alpha_i'(k)} \\ &= \frac{\alpha - \alpha_i'(k) - \gamma_{i,k}(1 - \alpha_i'(k))}{\alpha_i - \alpha_i'(k)} \\ &< 1 - \delta \cdot (1 - \alpha_i'(k)) \\ &\leq 1 - \delta \cdot (1 - \alpha_i) \\ &\leq 1 - \delta^2. \end{aligned}$$

The lemma now follows immediately by induction. □

Lemma 4.5.4. In Construction 4.5.1, for all $x \in \{0,1\}^*$,

$$k(x) \leq 1 + \frac{c|x|}{2\log e}$$

Proof. Fix $x \in \{0,1\}^*$. By Lemma 4.5.3 and the minimality of $k(x)$,

$$\alpha_i - (1-\delta^2)^{k(x)-1} \leq \alpha_i'(k(x)-1) < \alpha_i - (i+1)^{-2},$$

so

$$(1-\delta^2)^k(x) - 1 > (i+1)^{-2},$$

so

$$k(x) < 1 - \frac{2\log(i+1)}{\log(1-\delta^2)} \leq 1 + \frac{c|x|}{2\log e}.$$

□

Lemma 4.5.5. In Construction 4.5.1, for all $x \in \{0,1\}^*$,

$$n(x) \leq 2^{c|x|}.$$

Proof. Let $x \in \{0,1\}^*$. Then

$$n(x) = \sum_{k=0}^{k(x)-1} l(x,k),$$

so by Lemmas 4.5.2, 4.5.4, and the bound $1 + t \leq e^t$,

$$n(x) \leq \left(1 + \frac{c|x|}{2\log e}\right)^2 \leq e^{\frac{c|x|}{\log e}} = 2^{c|x|}.$$

□

Definition. Let (f,g) be a $\leq_{\mathrm{tt}}$-reduction.

1. (f,g) is *positive* (briefly, a $\leq_{\mathrm{pos-tt}}$-*reduction*) if, for all $A, B \subseteq \{0,1\}^*$, $A \subseteq B$ implies$F_{(f,g)}(A) \subseteq F_{(f,g)}(B)$.

2. (f,g) is *polynomial-time computable* (briefly, a $\leq_{\mathrm{tt}}^{\mathrm{P}}$-reduction) if the functions f and g are computable in polynomial time.

3. (f,g) is *polynomial-time computable with linear-bounded queries* (briefly, a $\leq_{\mathrm{tt}}^{\mathrm{P,lin}}$-*reduction*) if (f,g) is a $\leq_{\mathrm{tt}}^{\mathrm{P}}$-reduction and there is a constant $c \in \mathbf{N}$ such that, for all $x \in \{0,1\}^*$, $Q_{(f,g)}(x) \subseteq \{0,1\}^{\leq c(1+|x|)}$.

Of course, a $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reduction is a $\leq_{\mathrm{tt}}$-reduction with all the above properties.

The following result presents the properties of the positive bias reduction that are used in the proof of our main theorem.

Theorem 4.5.6 (Positive Bias Reduction Theorem). Let $\vec{\alpha}$ and $\vec{\beta}$ be strongly positive, P-exact sequences of biases, and let (f, g) be the positive bias reduction of $\vec{\alpha}$ to $\vec{\beta}$. Then (f, g) is an orderly $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reduction, and the probability measure induced by $\mu^{\vec{\beta}}$ and (f, g) is a coin-toss probability measure $\mu^{\vec{\alpha}'}$, where $\vec{\alpha} \approx \vec{\alpha}'$.

Proof. Assume the hypothesis. By inspection and Lemma 4.5.5, the pair (f, g) is an orderly $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reduction. (Lemma 4.5.5 also ensures that $f(x)$ is well-defined.) The reduction is also positive, since only AND's and OR's are used in the construction of $g(x)$. Thus (f, g) is an orderly $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reduction.

By remark (c) following Construction 4.5.1, the probability measure induced by $\mu^{\vec{\beta}}$ and (f, g) is the coin-toss probability measure $\mu^{\vec{\alpha}'}$, where $\vec{\alpha}' = (\alpha'_0, \alpha'_1, \ldots)$ is defined in the construction. Moreover,

$$\sum_{i=0}^{\infty} |\alpha_i - \alpha'_i| \leq \sum_{i=0}^{\infty} (i+1)^{-2} < \infty,$$

so $\vec{\alpha} \approx \vec{\alpha}'$. □

4.6 Equivalence for Complexity Classes

Many important complexity classes, including P, NP, co-NP, R, BPP, AM, P/Poly, PH, PSPACE, etc., are known to be closed under $\leq^{\mathrm{P}}_{\mathrm{pos-tt}}$-reductions, hence certainly under $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions. The following theorem, which is the main result of this chapter, says that the p-measure of such a class is somewhat insensitive to certain changes in the underlying probability measure. The proof is now easy, given the machinery of the preceding sections.

Theorem 4.6.1 (Bias Equivalence Theorem). Assume that $\vec{\alpha}$ and $\vec{\beta}$ are strongly positive P-sequences of biases, and let $\mathcal{C}$ be a class of languages that is closed under $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions. Then

$$\mu^{\vec{\alpha}}_{\mathrm{p}}(\mathcal{C}) = 0 \Longleftrightarrow \mu^{\vec{\beta}}_{\mathrm{p}}(\mathcal{C}) = 0.$$

Proof. Assume the hypothesis, and assume that $\mu^{\vec{\alpha}}_{\mathrm{p}}(\mathcal{C}) = 0$. By symmetry, it suffices to show that $\mu^{\vec{\beta}}_{\mathrm{p}}(\mathcal{C}) = 0$.

By Lemma 4.3.2, there exist P-exact sequences $\vec{\alpha}'$ and $\vec{\beta}'$ such that $\vec{\alpha} \approx \vec{\alpha}'$ and $\vec{\beta} \approx \vec{\beta}'$. Let (f, g) be the positive bias reduction of $\vec{\alpha}'$ to $\vec{\beta}'$. Then, by the Positive Bias Reduction Theorem (Theorem 4.5.6), (f, g) is an orderly $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reduction, and the probability measure induced by $\mu^{\vec{\beta}}$ and (f, g) is $\mu^{\vec{\alpha}''}$, where $\vec{\alpha}' \approx \vec{\alpha}''$.

Since $\vec{\alpha} \approx \vec{\alpha}' \approx \vec{\alpha}''$ and $\mu^{\vec{\alpha}}_{\mathrm{p}}(\mathcal{C}) = 0$, the Summable Equivalence Theorem (Theorem 4.2.3) tells us that there is a p-$\vec{\alpha}''$-martingale d such that $\mathcal{C} \subseteq S^{\infty}[d]$. By the Martingale Dilation Theorem (Theorem 4.4.3), the function $(f, g)\hat{\ }d$ is then a $\vec{\beta}'$-martingale. In fact, it easily checked that $(f, g)\hat{\ }d$ is a p-$\vec{\beta}'$-martingale.

Now let $A \in \mathcal{C}$. Then, since $\mathcal{C}$ is closed under $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions, $F_{(f,g)}(A) \in \mathcal{C} \subseteq S^{\infty}[d]$. It follows by the Martingale Dilation Theorem that $A \in S^{\infty}[(f, g)\hat{\ }d]$. Thus $\mathcal{C} \subseteq S^{\infty}[(f, g)\hat{\ }d]$. Since $(f, g)\hat{\ }d$ is a p-$\vec{\beta}'$-martingale, this shows that $\mu^{\vec{\beta}'}_{\mathrm{p}}(\mathcal{C}) = 0$. Finally, since $\vec{\beta} \approx \vec{\beta}'$, it follows by the Summable Equivalence Theorem that $\mu^{\vec{\beta}}_{\mathrm{p}}(X) = 0$. $\square$

It is clear that the Bias Equivalence Theorem remains true if the resource bound on the measure is relaxed. That is, the analogs of Theorem 4.6.1 for p_2-measure, pspace-measure, rec-measure, constructive measure, and classical measure all immediately follow. We conclude by noting that the analogs of Theorem 4.6.1 for measure in E and measure in E_2 also immediately follow.

Corollary 4.6.2. Under the hypothesis of Theorem 4.6.1,

$$\mu^{\vec{\alpha}}(\mathcal{C}|\mathrm{E}) = 0 \iff \mu^{\vec{\beta}}(\mathcal{C}|\mathrm{E}) = 0$$

and

$$\mu^{\vec{\alpha}}(\mathcal{C}|\mathrm{E}_2) = 0 \iff \mu^{\vec{\beta}}(\mathcal{C}|\mathrm{E}_2) = 0.$$

Proof. If $\mathcal{C}$ is closed under $\leq^{\mathrm{P,lin}}_{\mathrm{pos-tt}}$-reductions, then so are the classes $\mathcal{C} \cap \mathrm{E}$ and $\mathcal{C} \cap \mathrm{E}_2$.

□

CHAPTER 5 PARAMETER COMPLEXITY AND COMPUTATIONAL POWER IN QUANTUM COMPUTATION

5.1 Introduction to the Quantum Model

Following previous authors on the subject,*e.g.*, Bernstein and Vazirani [12], we introduce quantum Turing machines by first giving a description of probabilistic Turing machines and then transforming this description into the same language and notation that we will subsequently use for the quantum model. This description will differ significantly from the " distributions over witnesses" view that we used in Chapter 3. This alternative view begins with a nondeterministic Turing machine and assigns a probability to each transition. To keep the mechanism internally consistent we add the very natural requirement that from any given configuration the probabilities of the available transitions must sum to one. To progress from this starting point to a formalism more similar to that used with quantum Turing machines we will rephrase the model using linear algebra in a way analogous to that done with Markov processes.

5.1.1 A description of probabilistic Turing machines

Informally, we think of a probabilistic Turing machine as having a finite state control and a doubly infinite, linear storage tape. At each stage in the computation the read/write head reads one tape cell. Based on this symbol and its current state, the control selects a new state, writes a (possibly) new symbol at the current position on the tape and moves the read/write head one cell to the left (-1), keeps it where it is (0), or moves it one cell to the right (+1). To make the control mechanism probabilistic we allow the possibility that M might choose randomly from a number of alternative actions. Intuitively, the transition function $\delta((p,a),(q,b,d))$ gives the probability that for a given current state p and current tape symbol a, M will write the tape symbol b, change the state to q and move the tape head d cells to the right. We sometimes write this transition as $(p,a) \vdash (q,b,d)$. For a particular input pair (p,a) the values $\delta((p,a),(q,b,d))$ must sum to one.

A computation begins with the input string written on the tape, the read/write head scanning the left-most input symbol, and the control in state q_0. At each time step the machine selects one of the possible computation steps according to the probabilities assigned to them by δ. If the computation reaches state f^+ it stays there and we say M has accepted the input. If the computation reaches state f^- it stays there and we say that M has rejected the input.

We are usually interested in the probabilities with which these last two events happen. Due to the probabilistic nature of δ, there are potentially many possible computation sequences for a given input. The probability that any one specific computation sequence will occur is simply the product of the δ values of the individual transitions chosen in the sequence. The probability that M accepts an input x in t time steps is the sum of the probabilities of all the computation sequences of length t which find themselves in state f^+ after the t-th transition. Similarly, the probability that M rejects an input x in t time steps is the sum of the probabilities of all the computation sequences of length t which find themselves in state f^- after the t-th transition.

More formally we define probabilistic Turing machines as follows:

Definition.

A probabilistic Turing machine is a 6-tuple $M = (Q, \Sigma, \delta, q_0, f^+, f^-)$, whose components are as follows. $Q = \{q_0, q_1, \ldots, q_{n-2} = f^+, q_{n-1} = f^-\}$ is a finite set of states. Σ is a finite tape alphabet usually taken to be $\{0, 1, \Box\}$, where $\Box$ is the blank tape symbol. q_0 is the starting state. f^+ and f^- are the halt/accept and the halt/reject states respectively. δ is the transition function: $\delta : (Q \times \Sigma) \times (Q \times \Sigma \times \{-1, 0, +1\}) \longrightarrow [0, 1]$. We require that δ be well-formed, this means that at any point in the computation the probabilities of the possible transitions sum to one, *i.e.*, for any pair $(p, a) \in Q \times \Sigma$ we have

$$\sum_{(q,b,d) \in Q \times \Sigma \times \{-1,0,+1\}} \delta((p, a), (q, b, d)) = 1.$$

We also require that for any a, that $\delta((f, a), (f, a, 0)) = 1$ for $f = f^+$ or $f = f^-$. That is, once the machine achieves a final state nothing changes.

So far this has been a fairly standard description of probabilistic Turing machines. Next, we would like to develop a notation more like that used for quantum machines but which is still consistent with the above description of probabilistic machines.

There are two equivalent ways to view the semantics of a probabilistic Turing machine. The more usual view, that of the *realists*, says that at each time step the machine randomly chooses an actual transition using the probabilities given by δ. According to this interpretation the machine is in a definite configuration, *i.e.,* combination of state, head position and tape contents, at each time step. Realists view the machine as moving from one definite (but random) configuration to another.

The *observational* view, on the other hand, interprets the machine as having taken all possible paths simultaneously and being in a "superposition" of all possible configurations. An analogy might be to think of the condition of a single coin after it has been tossed but before the result has been observed. The realists' view would be that it is either a head or a tail with probability $\frac{1}{2}$ of each being the case. The observational view would be that the coin is in a state of being half heads and half tails simultaneously. Or to put it another way, the realists think it is either a head or a tail, we just do not know which yet, whereas the observationalists think it is really neither yet or more accurately that it is both until we look at it and "collapse" the possibilities to one definite result.

For probabilistic computation either view is equally valid. They both compute the same results with the same probabilities. For the quantum model, on the other hand, the two are not equivalent. The realists' view is inappropriate to this model. As we shall soon see, believing that the machine occupies a definite configuration during each intermediate step of a computation and computing conditional probabilities based on this belief will produce different results from those computed using the observational "superposition" viewpoint for quantum computations. This counterintuitive state of affairs distinguishes quantum computation from classical or probabilistic computation in the same way that it distinguishes quantum physics from classical physics.

To best understand the processes in quantum computation it will be convenient to formalize the notion of a configuration of a Turing machine.

Definition.

A *configuration* of a Turing machine $M = (Q, \Sigma, \delta, q_0, f^+, f^-)$ is a 4- tuple, $< m, w_l, a, w_r > \in \{0, 1, \ldots, |Q| - 1\} \times \{0,1\}^* \times \{0, 1, \Box\} \times \{0,1\}^*$. In this 4-tuple, q_m is the current state, a the tape symbol currently under the read/write head, and w_l, and w_r represent the (possibly empty) strings to the left and right, respectively, of the tape head.

We define the set of all configurations as $config_M = \{0, \ldots, |Q| - 1\} \times \{0,1\}^* \times \{0, 1, \Box\} \times \{0,1\}^*$.

From a configuration $< m, w_l, a, w_r >$ the transition $(q_m, a) \vdash (q_n, b, d)$ will take us to a new configuration give by:

$$< n, w_l[0 \ldots |w_l| - 2], w_l[|w_l| - 1], b \cdot w_r > \quad \text{if } d = -1$$

$$< n, w_l, b, w_r] > \quad \text{if } d = 0$$

$$< n, w_l \cdot b, w_r[1], w_r[2 \ldots |w_r| - 1] > \quad \text{if } d = +1.$$

The set $config_M$ is countable, so we may write $config_M = \{c_0, c_1, \ldots\}$. We can then represent the set of configurations possible after t time steps of machine M with input x as a vector $\vec{v}_M^{(t)}(x) =< v_0, v_1, \ldots >$ where the ith component represents the probability that machine M is in configuration c_i at time t. It will be the case that all but finitely many of the components of $\vec{v}_M^{(t)}(x)$ will be zero, that all of them will be in the interval $[0, 1]$, and since δ is well-formed, that they will sum to one. We will suppress the M subscript parameter and write $\vec{v}^{(t)}(x)$ when an M has been fixed or is evident. If both an M and an x have been fixed we will write $\vec{v}^{(t)}$ for $\vec{v}_M^{(t)}(x)$. This will unclutter the notation.

The last step in the transition to a matrix notation is to transform the function δ into a matrix. Let D be the matrix whose (i, j)th entry $D_{i,j}$ is the probability that M in configuration c_i moves to configuration c_j. Then it is clear that, although D is infinite in two dimensions, each row and column has at most finitely many non-zero entries. It is also clear that each row sums to one. From our interpretation of vector components as configuration probabilities it is easy to see that if we multiply a row vector whose entries sum to one by D (on the right) the result is another row vector whose entries sum to one. Not all matrices that satisfy these constraints represent transition functions of Turing machines so we call D *regular* if there is a probabilistic Turing machine M with transition function δ such that D corresponds to δ in the way just described. When D corresponds to transition function δ, and hence is regular, we write $D = D(\delta)$.

We are now in a position to describe a probabilistic computation in terms of matrix multiplication. Fixing a probabilistic Turing machine $M = (Q, \Sigma, \delta, q_0, f^+, f^-)$, we let $init(x) =< 0, \lambda, x[0], x[1 \ldots |x| - 1] >$ be the initial configuration of M on input x. This says that x has been written on the work tape, that the read/write head is positioned over the left-most bit of x and that the machine control is in state 0. We let $\vec{v}^{(0)}(x)$ be the vector with a one in the position corresponding to $init(x)$ and zeroes elsewhere. Then each step of the computation $M(x)$ can be represented by multiplication by D. Thus, if we set $\vec{v}^{(t)}(x) = \vec{v}^{(0)}(x) \cdot D^t$, then $\vec{v}^{(t)}(x)[i]$ represents the probability that M finds itself in configuration c_i after t time steps when computing beginning with configuration $init(x)$.

We end this section by defining the semantics of accepting and rejecting inputs. Recall that f^+ is the accept state and that f^- is the reject state. To find the probability that M accepts an input string x after t time steps we need to sum the components in $\vec{v}^{(t)}(x)$ which correspond to configurations of M in the state f^+. Similarly we can compute the probability of rejection using the state f^-. More formally:

<u>Definition</u>.

For an n-state probabilistic Turing machine $M = (Q, \Sigma, \delta, q_0, q_{n-2} = f^+, q_{n-1} = f^-)$ we define the set of accepting configurations as

$$acc(M) = \{n-2\} \times \{0,1\}^* \times \Sigma \times \{0,1\}^*$$

and the set of rejecting configurations as

$$rej(M) = \{n-1\} \times \{0,1\}^* \times \Sigma \times \{0,1\}^*.$$

The probability that M accepts input string x in t time steps is denoted by $p_+^{(t)}(M,x)$ and the probability that M rejects input string x in t time steps by $p_-^{(t)}(M,x)$. Their values are given by $p_+^{(t)}(M,x) = \sum_{c_i \in acc(M)} v^{(t)}(x)[i]$ and $p_-^{(t)}(M,x) = \sum_{c_i \in rej(M)} v^{(t)}(x)[i]$.

5.1.2 Some polynomial-time probabilistic complexity classes

By restricting machines to certain levels of acceptance and by limiting the range of δ we can define the various well-known probabilistic complexity classes. A reader who is primarily interested in continuing the discussion of the quantum model can skip this section.

To define the usual classes RP, ZPP, and BPP we require that the range of δ be the set $\{0, \frac{1}{2}, 1\}$. That is, each step in the computation has at most two choices and when two choices are available they are equally likely to be chosen. Under this assumption we can define the classes just listed.

Definition.

Let $M = (Q, \Sigma, \delta, q_0, f^+, f^-)$ be a probabilistic Turing machine. We call δ a *fair-coin transition function*, and M a *fair-coin probabilistic Turing machine* if $range(\delta) = \{0, \frac{1}{2}, 1\}$.

Definition.

A language $L \subseteq \{0,1\}^*$ is in the class RP if there exist a polynomial q and a fair-coin probabilistic Turing machine M such that for $x \in L$ we have $p_+^{(q(|x|))}(M, x) > \frac{1}{2}$ and for $x \notin L$ we have $p_+^{(q(|x|))}(M, x) = 0$.

A language $L \subseteq \{0,1\}^*$ is in the class ZPP if there exist a polynomial q and a fair-coin probabilistic Turing machine M such that for $x \in L$ we have $p_+^{(q(|x|))}(M, x) > \frac{1}{2}$ and that $p_-^{(q(|x|))}(M, x) = 0$ while for $x \notin L$ we have $p_-^{(q(|x|))}(M, x) > \frac{1}{2}$ and that $p_+^{(q(|x|))}(M, x) = 0$.

A language $L \subseteq \{0,1\}^*$ is in the class BPP if there exist a polynomial q and a fair-coin probabilistic Turing machine M such that for $x \in L$ we have $p_+^{(q(|x|))}(M, x) > \frac{3}{4}$ and for $x \notin L$ we have $p_-^{(q(|x|))}(M, x) > \frac{3}{4}$.

This definition of BPP is equivalent to the definition given in Chapter 3.

To redefine the class $\mathrm{BP}_\beta\mathrm{P}$ discussed in Chapter 3 we require that the M have at most 2 choices in any given configuration and that when M has exactly 2 choices, one has probability β and the other $1 - \beta$. The acceptance criterion is the same as that just given for BPP.

Definition.

Let $M = (Q, \Sigma, \delta, q_0, f^+, f^-)$ be a probabilistic Turing machine. We call δ a *biased-coin transition function*, and M a *biased-coin probabilistic Turing* machine if there is a real number β such that $range(\delta) = \{0, \beta, 1-\beta, 1\}$ and for every $(q, a) \in Q \times \Sigma$ there are at most 2 choices of $(p, b, d) \in Q \times \Sigma \times \{-1, 0, 1\}$ for which $\delta((q, a), (p, b, d)) > 0$.

It is a consequence of the definition of probabilistic Turing machines that any such β will be in the interval $[0, 1]$.

Definition.

A language $L \subseteq \{0,1\}^*$ is in the class $\mathrm{BP}_\beta\mathrm{P}$ if there exist a polynomial q and a biased-coin probabilistic Turing machine M having bias β such that for $x \in L$ we have $p_+^{(q(|x|))}(M, x) > \frac{3}{4}$ and for $x \notin L$ we have $p_-^{(q(|x|))}(M, x) > \frac{3}{4}$.

This is equivalent to our previous definition of $\mathrm{BP}_\beta\mathrm{P}$, and as before, $\mathrm{BPP} = \mathrm{BP}_{\frac{1}{2}}\mathrm{P}$.

5.1.3 The quantum model

Having described the probabilistic model we are ready to describe the quantum model in similar terms. In our definition a quantum Turing machine will consist of a set of states, use the binary alphabet, have a complex-valued transition function, and have a set of halting states (some accepting and some rejecting).

To represent the current state we will use a string of bits, which we can interpret as a natural number or, if one prefers, as a register. Rather than having a single accept state, it will be convenient to think of one of the bits as representing acceptance/rejection. It will also be convenient to use one of the state bits to 'signal' when the computation is complete. We will discuss the importance of this bit later, its necessity and use is one of the fundamental differences between probabilistic and quantum Turing machines.

The transition function of the quantum Turing machine will take complex values rather than be limited to the interval $[0, 1]$ of reals. This use of complex values will have very serious and interesting consequences for our understanding and interpretation of the computational process. The transition function will be converted into a transition matrix in the same way we transformed the transition function of a probabilistic machine into matrix form. We can again then think of each step in a computation as a matrix multiplication. One significant difference however will be that we will require that the matrix be unitary, *i.e.*, $D^{-1} = D^*$. This means that only transition functions that produce unitary transition matrices will be allowed.

Because of the complex-valued nature of the transition functions we will not be able to directly interpret their values as probabilities. Using the language and usage of quantum physics, we will instead call these complex values amplitudes. The associated probabilities will come from the squares of the moduli of the amplitudes. Thus, we will still be able to think of the values as giving the probabilities of transitions and the probabilities of being in specified states, albeit indirectly.

We now give a formal definition of a quantum Turing machine.

Definition.

A *quantum Turing machine* is a 5-tuple $Q = (S, \Sigma, s_0, \delta, F)$, where S is a finite set of states, Σ is a tape alphabet, $s_0 \in S$ is the starting state, $\delta : (S \times \Sigma) \times (S \times \Sigma \times \{-1, 0, 1\}) \to \mathbf{C}$ is a regular transition function, and $F \subseteq S$ is a set of accepting states.

We say that δ is *regular* if the associated transition matrix $D(\delta)$ is unitary.

We canonically use $S = \{0, 1, \ldots 2^k - 1\}$, $\Sigma = \{0, 1, \Box\}$, $s_0 = 0$, and $F = \{n | \text{the } k - 2\text{nd bit of } n \text{ is } 1\}$

The state of Q is represented by a k-bit string (or register). The most significant (leftmost) bit is interpreted as the halt flag and the next most significant bit as the accept flag. To accept an input a computation will raise both flags. To reject an input it will raise only the halt flag. In a discussion to follow we will see that a quantum Turing machine cannot really halt. It will raise the halt flag and continue processing. However, we will not allow it to change its mind about acceptance or halting.

As with probabilistic Turing machines we note that the set of configurations is countable, we denote the set of all configurations of a quantum Turing machine Q as $config_Q$ and enumerate the set as $\{c_0, c_1, ...\}$. We think of the computation in terms of beginning with an initial vector $\vec{v}^{(0)}(x)$ having a one in the component corresponding to the initial configuration $init_Q(x)$ and zero for all other components. This vector is subsequently multiplied by the matrix $D(\delta)$, once for each step of the computation, producing, after t time steps, a complex-valued vector $\vec{v}^{(t)}(x)$ whose i-th component v_i is the amplitude associated with the configuration c_i. The squared modulus of v_i, $|v_i|^2$, gives the probability that the computation will be in configuration c_i after the t-th time step. We desire that the total probability of the various configurations should be one at all times. That is trivially true for the initial configuration vector. We must restrict $D(\delta)$ such that it preserves this property. One class of matrices with this property is the class of unitary matrices. In fact if a square matrix D defines an onto mapping then it is unitary if and only if it preserves length. So unitaricity is a very natural requirement for quantum transition matrices.

The definitions below are standard definitions from complex analysis and linear algebra.

Definition.

The *modulus* of a complex number $z = a + \imath b$ is given by $|z| = \sqrt{a^2 + b^2}$.

The *norm* or *length* of a (configuration) vector v is given by $\|v\| = \sum_{i=0}^{\infty} |v|^2$.

There are many equivalent ways of defining unitary matrices. Our earlier definition was that D is unitary if $D^{-1} = D^*$. Another equivalent characterization is that if D defines an onto mapping then it is unitary if and only if it preserves length, *i.e.*, $\|\vec{v}D\| = \|\vec{v}\|$. Both of these definitions have the drawback that we cannot tell directly from the transition function whether or not it yields a unitary transition matrix. In a subsequent section we will look at a "local" characterization of the unitary concept given by Bernstein and Vazirani [12]. First however, we examine the consequences of allowing complex amplitudes.

There are two important consequences of allowing quantum machines to have complex (or even negative real) amplitudes. The first is interference, *i.e.,* the ability for computation paths to reinforce each other or to cancel each other. These are called constructive and destructive interference respectively. The second is the effect that an observer can have on the result of a computation. The example below demonstrates both phenomena. It is a commonly used [18, 70], simple example of quantum computation.

Consider a quantum Turing machine in which there are only two configurations. This can be thought of as a single bit, often called a *qbit*, which is either 0 or 1. Consider the transition matrix

$$D(\delta) = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}.$$

Intuitively this may be thought of as the transition matrix for a fair quantum coin. The configuration vector $< 1, 0 >$ may be though of as corresponding to heads and $< 0, 1 >$ to tails. If the machine starts in the $< 1, 0 >$ configuration and we apply $D(\delta)$ once we obtain $< \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} >$. Interpreting the squares of the components as probabilities, we see that we are in a superposition of configurations which is 1/2 heads and 1/2 tails. This is the basis of our claim that this corresponds to a fair coin. To see the interference, let us apply $D(\delta)$ again. Multiplying $< \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} >$ by $D(\delta)$ returns us to $< 1, 0 >$. We were in a condition of being half heads and half tails, the amplitude of moving from heads to tails was $1/\sqrt{2}$ which yields probability $\frac{1}{2}$. Similarly, the amplitude of moving from tails to tails was $-1/\sqrt{2}$ which yields probability $\frac{1}{2}$. So from heads to tails has positive probability, and from tails to tails has positive probability, and yet from a state of being half heads and half tails our total probability of moving to tails is zero! What happened?

The key to this effect lies in the negative amplitude. The negative amplitude of the tails-to-tails transition cancels out the positive amplitude of the heads-to-tails transition so that there remains no probability of getting a tail.

This interference is more than just a curiosity. Well-designed quantum algorithms use interference to great effect. They are designed so that undesirable outcomes suffer destructive interference while desirable outcomes are enhanced by constructive interference.

We can use the same example to demonstrate the power of observation. When we *observe* a quantum (or probabilistic) computation after the t-th time step, we must see the machine Q in one of the legal configurations in $config_Q$. It cannot be an abstract superposition of configurations such as exist in the absence of observation. That is, the physical coin, when observed, must either be heads or tails. Quantum physicists call this effect 'collapsing the quantum possibilities' to a classical result. We saw in the previous example that if we start the coin at heads and flip it twice without looking at the intervening result the result is always a head. Suppose we *observe* the coin after the first toss. This forces the coin into configuration $< 1, 0 >$ or $< 0, 1 >$ with probabilities $\frac{1}{2}$ and $\frac{1}{2}$ respectively. Using either of these configurations as a starting point the second coin toss results in a coin that is heads with probability $\frac{1}{2}$, whereas the unobserved coin was heads with probability 1. When we do not observe the intermediate result we always get a head. When we do observe the intermediate result we get a head half of the time and a tail half of the time. The observation act has changed the distribution of the outcome!

In the probabilistic case an observation may limit the outcomes for that run of the computation but the distribution of outcomes over many runs is unchanged. On the other hand an observation of a quantum computation may actually change the long-run distribution of the outcomes.

To avoid this effect we do not allow ourselves to observe a quantum computation in progress. We only allow ourselves to *observe* the halt bit. We assume our computation is not sensitive to this particular observation. Only when the halt flag has been raised do we observe the result of the computation.

The previous fair-coin example can be modified to model a biased coin. Applying the transition matrix below to the configuration vector $< 1, 0 >$ yields $< \sqrt{\beta}, \sqrt{1-\beta} >$. Such a vector may be taken to represent the result of tossing a coin with bias β.

$$D(\delta) = \begin{bmatrix} \sqrt{\beta} & \sqrt{1-\beta} \\ \sqrt{1-\beta} & -\sqrt{\beta} \end{bmatrix}$$

In both examples since $D(\delta) = D^*(\delta) = D^{-1}(\delta)$ we have that $D(\delta)$ is unitary. We will be able to extend this biased-coin example to achieve quantum results related to the $\mathrm{BP}_\beta\mathrm{P}$ classes that were the main topic of Chapter 3. But first we look more closely at the problem of deciding whether or not a quantum transition function is regular.

Unlike the two simple examples just given, checking that $D(\delta)$ is unitary is usually a difficult task. We would like a 'local' method, *i.e.*, one that looks only at δ, for determining whether δ is regular. The following theorem from Bernstein and Vazirani [12, 79] is one such useful characterization of regular transition functions.

Theorem 5.1.1.(Bernstein and Vazirani)[79]

The quantum transition function δ for $Q = (S, \Sigma, s_0, \delta, F)$ is regular if and only if the following two conditions hold.

1. For any $(p_1, a_1), (p_2, a_2) \in S \times \Sigma$,

$$\sum_{(q,b,d) \in S \times \Sigma \times \{-1,0,1\}} \delta((p_1, a_1), (q, b, d)) \delta^*((p_2, a_2), (q, b, d))$$

$$= \begin{cases} 1 & \text{if } (p_1, a_1) = (p_2, s_2) \\ 0 & \text{if } (p_1, a_1) \neq (p_2, s_2) \end{cases}$$

2. For any $(p_1, a_1, b_1), (p_2, a_2, b_2) \in S \times \Sigma \times \Sigma$ and for any $d_1 \neq d_2 \in \{-1, 0, 1\}$,

$$\sum_{q \in S} \delta((p_1, a_1), (q, b_1, d_1))\delta^*((p_2, a_2), (q, b_2, d_2)) = 0.$$

The first condition clearly comes from $DD^* = I$. The second says that the various ways to get to a particular state are orthogonal. The later in turn is related to a fundamental property of quantum Turing machines, namely, their reversibility.

In quantum physics, one of the consequences of the unitary nature of the matrices is that quantum processes are reversible. The matrix D^* gives the amplitudes associated with reversing the computation of a quantum Turing machine. In fact, Bernstein and Vazirani [12] tell us that this is another way to characterize regular transition functions.

Theorem 5.1.2.(Bernstein and Vazirani)[12]

The transition function δ of a quantum Turing machine is regular if and only if the configuration map δ^* specified by reversing the arrows and conjugating the amplitudes of δ undoes the computation of δ.

Bennett [6, 7, 8] has shown that any deterministic Turing machine is reversible. The theorem above tells us that reversible Turing machines form a subset of the quantum Turing machines. Thus quantum Turing machines are a generalization of classical Turing machines. This observation was made by Deutsch in his seminal paper on quantum computation[25]. Furthermore, the Bennett results [6, 7, 8] tell us that the transformation from classical to reversible Turing machine causes only a linear slowdown in the running times. Thus the simulation of deterministic algorithms by quantum machines is efficient. We will use these results on the quantum simulation of classical machines in the next sections.

5.2 The Complexity of Quantum Parameters

In Chapter 3 we showed that the complexity of the parameter β was intricately related to the complexity of the class $\mathrm{BP}_\beta\mathrm{P}$. In particular, we showed that for some β, $\mathrm{BP}_\beta\mathrm{P}$ contains noncomputable languages. This was true because we could recover noncomputable information encoded in the bias β.

We will show that a similar phenomenon is true for quantum complexity classes. The quantum parameters in question are the values assumed by the transition function δ which in turn are the entries of $D(\delta)$. If they are computable then a deterministic algorithm can perform the matrix multiplications needed to calculate the transitions of the quantum machine to necessary accuracy with at most exponential slowdown. Thus, a quantum Turing machine with a computable transition function decides a computable language.

We will show, on the other hand, that there are quantum parameter sets which can compute noncomputable languages. The type of parameters allowed is a resource to the computation just as space and time are resources. Restricting this resource produces a hierarchy of complexity classes just as restricting space and time resources produce hierarchies. In the next section we will define quantum classes based on the complexity of these parameters.

The fact that useful information can be encoded in the values of $D(\delta)$ is a technical point that was overlooked in early papers on quantum computation. These papers state that quantum computers are limited to deciding computable languages because the matrix computations can be simulated deterministically, albeit with exponential slowdown. This line of reasoning however implicitly assumes that the values of the parameters can be obtained to any desired accuracy upon demand. This tacit assumption limits the parameters to be "computable" complex numbers. This assumption breaks down even further when one enters the realm of time-bounded quantum computation. Here even the assumption that the parameters are computable gives an algorithm unreasonable computational power. We believe that the parameters should be computable in time compatible with the quantum algorithm.

Recently authors have been more careful about these implicit assumptions. For example, Shor [67] states that it is desirable for quantum machines to have polynomial precision in their amplitudes. *i.e.,* For an input of length n one needs to specify the first $\log n$ bits of the amplitudes.

In the next section we will adopt the convention of explicitly stating the set of allowed amplitudes when defining a quantum complexity class. We would write Shor's suggestion as BQP[E_{CF}].

A theorem given by Bennett, Brassard, Bernstein and Vazirani [9] quoted in section 5.3 also says that we only need to know the values of the amplitudes to an accuracy of $2^{-O(\log n)}$, where n is the length of the input, to determine the result of a quantum computation. Thus, we believe that BQP[E_{CF}] is the best choice for a "standard" BQP.

5.3 Quantum Complexity Classes

We have developed our view of quantum computation by analogy to probabilistic computation. We also define the semantics of acceptance to parallel to those of probabilistic computation, in particular, we define the quantum complexity class BQP[T] in a way similar to the way we defined the class BPP.

Definition.

Let T be a set of complex numbers and L be a language. We say that L is a member of BQP[T] if there exist a quantum Turing machine $Q = (S, \Sigma, s_0, \delta, F)$ and a polynomial q such that $range(\delta) \subseteq T$ and for every x, after $q(|x|)$ time steps, all computation paths of Q have raised the halt flag and for $x \in L$ the probability the Q accepts is at least $\frac{3}{4}$ and for $x \notin L$ the probability that Q accepts x is at most $\frac{1}{4}$.

5.3.1 A relationship between $\mathrm{BP}_\beta\mathrm{P}$ and BQP

One of the ways we could approach the relationship between probabilistic classes and quantum classes would be based on the definition of BQP[T] in the previous section. We could start with a probabilistic Turing machine, extend Bennett's reversibility results to probabilistic Turing machines and proceed to establish a containment of probabilistic classes in quantum classes.

However, we prefer to return at this point to the probability over witnesses definition of $\mathrm{BP}_\beta\mathrm{P}$ that we gave in Chapter 3. This will allow us to use deterministic Turing machines and the Bennett/Deutsch results directly. In the theorem below, we will use the biased-coin transition to create a string of β-biased bits. This will produce a string which represents a superposition of all possible witnesses in which each possible witness w appears with probability $\mu^\beta(w)$. We can then simulate the usual deterministic algorithm on the input-witness pair.

Theorem 5.3.1.

$$\mathrm{BP}_\beta\mathrm{P} \subseteq \mathrm{BQP}[\{0, \sqrt{\beta}, \sqrt{1-\beta}, 1\}].$$

Proof.

Consider a language L in $\mathrm{BP}_\beta\mathrm{P}$. There must be a language B in P and a polynomial $q(n)$ such that for any x

$$Pr_{\mu^\beta}(\{w \in \{0,1\}^{q(|x|)} | < x, w > \in B \iff x \in L\}) > \frac{3}{4}.$$

By Bennett's work there is a reversible Turing machine M which accepts B.

We now define a BQP algorithm Q which will recognize L.

The algorithm for Q first(reversibly) writes a string of $q(|x|)$ many ones on the work tape. It then applies the quantum β-biased-coin transition independently to each of these bits. This produces a string w which is a superposition of all possible $q(|x|)$-length strings having probability distribution μ^β.

The machine Q deterministically (and reversibly) creates the pair $< x, w >$ and simulates $M(< x, w >)$. Since the witnesses were distributed according to μ^β, the acceptance probability of Q will be greater than $\frac{3}{4}$ for $x \in L$ and less than $\frac{1}{4}$ for $x \notin L$.

The values in $D(\delta)$ for Q are exactly the $\sqrt{\beta}$ and $\sqrt{1-\beta}$ needed for the biased coin flips and $\{0, 1\}$ for the deterministic steps.

The transition function is reversible thus by Theorem 5.1.2 we have that it is regular and is a BQP$[\{0, \sqrt{\beta}, \sqrt{1-\beta}, 1\}]$ machine and the proof is complete.

□

Using $\beta = \frac{1}{2}$ we easily get BPP $\subseteq$ BQP$[\{0, \frac{1}{\sqrt{2}}, 1\}]$. In addition our BP$_\beta$P results from Chapter 3 also carry over. The following corollary states two of these.

Corollary 5.3.2.

- If $\beta \in \mathrm{E_{CF}}$ then BP$_\beta$P $\subseteq$ BQP.
- There is a set T of such that BQP$[T] \not\subseteq$ REC.

Proof.

The first inclusion holds since we can easily compute the square root of a number given exponential time.

In Chapter 3 we showed that for some β, BP$_\beta$P contained noncomputable languages. If L is one such noncomputable element of some BP$_\beta$P then by Theorem 5.3.1 there is a quantum Turing machine in the class BQP$[\{0, \sqrt{\beta}, \sqrt{1-\beta}, 1\}]$ which accepts L. The second statement of the corollary follows.

□

5.3.2 $\mathrm{BQP}[\mathrm{E_{CF}}] \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$

We continue our investigation of quantum complexity classes by showing that $\mathrm{BQP}[\mathrm{E_{CF}}] \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$, *i.e.*, if the amplitudes associated with a quantum Turing machine Q are exponential-time computable then the language accepted by Q can be simulated by a polynomial time-bounded Turing machine with a #P oracle. This result is a more precise and slightly strengthened version (with a new proof) of a result that Bernstein and Vazirani [12] attribute to personal communication by Valiant.

We remind the reader of the definition of the counting class #P, first introduced by Valiant [73].

Definition.

If M is a nondeterministic Turing machine with polynomial time bound $q(n)$ such that M is balanced, *i.e.*, every computation takes exactly $q(|x|)$ steps, then the function, $h : \{0,1\}^* \to \mathbf{N}$, defined by $h(x)$ = the number of distinct accepting computations of M(x), is called the *counting function associated with M*.

The collection of all of the counting functions for all such NTMs is denoted by #P and called *sharp* P.

Since we can rewrite any Turing machine algorithm to clock itself and terminate at a precomputed time, the balance restriction in the above definition does not limit the class #P.

We use three lemmas in the proof of the main theorem of this section. The first is from Bernstein and Vazirani [12]. The second is from Bennett, Bernstein, Brassard and Vazirani [9]. The third is original and contains the crux of the argument.

Lemma 5.3.3.[12]

To every quantum Turing machine Q there corresponds another quantum Turing machine Q' whose transition function δ assumes only real values and such that

$$Pr(Q \text{ accepts } x) = Pr(Q' \text{ accepts } x).$$

Lemma 5.3.3 is stated without proof in [12]. The proof can be obtained by splitting states into real and imaginary parts, building a Cartesian product machine and carefully defining the transition function to properly preserve complex arithmetic operations.

Lemma 5.3.4.[9]

For any polynomial time bounded quantum Turing machine Q there is a constant c such that for all x, if any other quantum Turing machine Q' has the same state set, same tape alphabet and satisfies $|\delta(s_1, \sigma_1, \sigma_2, s_2, d) - \delta'(s_1, \sigma_1, \sigma_2, s_2, d)| \leq 2^{-c \cdot \log(|x|)}$ for all $(s_1, \sigma_1, s_2, \sigma_2, d)$, then Q accepts x if and only if Q' accepts x.

The previous lemma was discussed in Section 5.2. As stated then, we believe that this lemma exhibits a critical property of quantum computation.

The next lemma contains the crux of the argument that $\mathrm{BQP}[\mathrm{E_{CF}}] \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$. In this lemma we construct a nondeterministic Turing machine whose counting function will act as oracle in the proof of the main theorem. The paths in the machine encode the amplitudes of Q using a two's complement representation. We recall for the reader the definition of the two's complement representation.

Definition.

The *k-bit, two's complement representation of an integer z* is the function tc_k : $\{-2^{k-1}, \ldots, 2^{(k-1)} - 1\} \rightarrow \{0,1\}^k$ defined as follows.

1. If $0 \leq z < 2^{(k-1)}$ then $tc_k(z)$ is the standard binary representation of z padded on the left with enough zeros to extend the string to exactly k-bits.

2. If $-2^{(k-1)} \leq z < 0$ then then $tc_k(z)$ is the standard binary representation of $2^k + z$ which will be exactly k-bits long.

This representation is important because for operands and results in the specified domain and range it preserves the basic arithmetic operations. With this domain and range it is one-to-one and onto, and is therefore invertible. In the theorem we use this inverse somewhat loosely by left-truncating longer arguments and left-padding shorter arguments to make exactly k bits before applying $tc_k^{-1}()$ to elements of $\{0,1\}^*$.

Lemma 5.3.5.

If Q is a quantum Turing machine with polynomial time-bound $q(n)$, and if the values for δ are computable to accuracy $2^{-2c\cdot\log(n)}$ in time polynomial in n, then there exist an NTM N and a polynomial g(n) such that for all x,

$$tc_{g(|x|)}^{-1}(\#acc(N,x)) = 2^{2c\cdot\log(|x|)} \cdot Pr(Q \text{ accepts } x).$$

Proof.

Let Q, q and c satisfy the hypotheses. By Lemma 5.3.3 we may assume that the transition function of Q assumes real values. We first will define a nondeterministic Turing machine N. The intuition behind N is that we take a computation tree for Q and 'integerize' it by multiplying the amplitudes by a suitably large power of two and rounding. Since some of the transition amplitudes may be negative we use a two's complement encoding of these integers of a sufficient length, $g(n)$, so that no overflow occurs when we multiply $q(n)$ of these amplitudes together. An arc in the computation tree of Q with (now) unsigned integer amplitude m is replicated in N m times along with the subtree to which it leads so that as we proceed down the computation tree, the number of paths in N encodes the product of the amplitudes along the corresponding path in Q.

The algorithm for N is as follows.

begin N(x)

1. Compute the values of δ, and hence also $D(\delta)$ to precision $2^{-c\cdot\log(|x|)}$.

2. Convert the values from step 2 to integers by multiplying by $2^{c \cdot \log(|x|)}$ and rounding. Call these values δ' and $D'(\delta)$. Let $n(|x|)$ be maximum of the bit lengths of the absolute values of these integers. Define $g(|x|) = (n(|x|)+1)q(|x|)$.

3. Convert the values from step 2 to two's complement using a $g(|x|)$-bit representation. Call these values δ'' and $D''(\delta)$.

4. Nondeterministically select an accepting configuration e for $Q(x)$.

5. **for** $k = 1$ **to** 2 **do** SubQ$(e, q(|x|), k)$

end (N)

Intuitively, steps 1 through 3 compute the two's complements of the 'integerized' amplitudes. Step 4 will allow us to perform a simple sum over the accepting configurations and step 5 will square the number paths since we trace from accepting configuration to initial configuration twice. The SubQ procedure below traces a path from accepting state back to the initial state in such a way that the number of paths encodes the product of the amplitudes along such a path. Recall that the length function g is long enough so that we can multiply $q(n)$-many n-bit numbers without overflowing the encoding.

begin SubQ(e_i, t, k);

1. **if** $t = 0$ and $k = 1$ and $e_i = init_Q(x)$ **then return**

2. **if** $t = 0$ and $k = 2$ and $e_i = init_Q(x)$ **then accept**

3. **if** $t = 0$ and $e_i \neq init_Q(x)$ then **reject**

4. **if** $t \neq 0$ **then**

   (a) Nondeterministically select a configuration e_j such that $D''_{ji} \neq 0$. *i.e.*, an e_j which can reach e_i in a single transition.

(b) Nondeterministically select from D''_{ji} many identical branches each of which executes SubQ$(e_j, t-1, k)$

end SubQ.

Step 1 of SubQ returns us from the subroutine to perform the second iteration of the loop in step 5 of algorithm N. Steps 2 and 3 of SubQ match accepting configurations of Q with accepting paths in N and step 4 multiplies the number of paths in N so their number encodes the product of the amplitudes along the corresponding path in Q.

By $\#paths(e,f,t)$ we mean the number of paths in N corresponding to a computation from configuration e to configuration f of length t in Q.

We claim that the number of such paths in N satisfies

$$tc^{-1}_{g(|x|)}(\#paths(init_Q(x), e_i, t)) = 2^{c \cdot t \cdot \log(|x|)} \cdot \vec{v}^{(t)}(x)[i].$$

We prove this by induction on t.

If $t = 0$ then the quantum Turing machine is in the initial configuration, $init_Q(x)$. Hence, there is exactly one path, namely the empty path, from $init_Q(x)$ to e_i exactly when $e_i = init_Q(x)$. There is no path otherwise. Further, $\vec{v}^{(0)}[i] = 1$ exactly when $e_i = init_Q(x)$ and 0 otherwise so the claim is true for $t = 0$.

Assume that the claim holds for $t-1$, *i.e.*, suppose that

$$tc^{-1}_{g(|x|)}(\#paths(init_Q(x), e_i, t-1)) = 2^{c(t-1)\log(|x|)} \cdot \vec{v}^{(t-1)}(x)[i].$$

Then

$$\begin{aligned} & tc^{-1}_{g(|x|)}(\#paths(init_Q(x), e_i, t)) \\ &= tc^{-1}_{g(|x|)}\left(\sum_j D''_{ji} \cdot \#paths(init_Q(x), e_j, t-1)\right) \\ &= tc^{-1}_{g(|x|)}\left(\sum_j tc_{g(|x|)}\left(D_{ji} \cdot 2^{c \cdot \log(|x|)}\right) \cdot \#paths(init_Q(x), e_j, t-1)\right) \end{aligned}$$

$$\begin{aligned}
&= \sum_j D_{ji} \cdot 2^{c\cdot\log(|x|)} \cdot 2^{c\cdot(t-1)\cdot\log(|x|)} \cdot \vec{v}^{(t-1)}(x)[i] \\
&= 2^{ct\cdot\log(|x|)} \cdot \sum_j D_{ji} \cdot \vec{v}^{(t-1)}(x)[i] \\
&= 2^{ct\cdot\log(|x|)} \cdot \vec{v}^{(t)}(x)[i].
\end{aligned}$$

Thus by induction the first claim is proven.

Using this claim we complete the proof of this lemma counting the number of accepting paths for N.

$$\begin{aligned}
&tc^{-1}_{g(|x|)}(\#acc(N,x)) \\
&\quad = tc^{-1}_{g(|x|)}\left(\sum_{e_i \text{ accepting}} (\#paths(init_Q(x), e_i, q(|x|)))^2\right) \\
&\quad = \sum_{e_i \text{ accepting}} \left(tc^{-1}(\#paths(init_Q(x), e_i, q(|x|)))\right)^2 \\
&\quad = \sum_{e_i \text{ accepting}} \left(\left(2^{c\cdot q(|x|)\cdot\log(|x|)}\right)^2 \cdot \left(\vec{v}^{(q(|x|))}(x)[i]\right)^2\right) \\
&\quad = 2^{2c\cdot q(|x|)\cdot\log(|x|)} \cdot Pr(Q \text{ accepts } x)
\end{aligned}$$

Thus the lemma is proven. □

With these lemmas we can prove the following theorem.

Theorem 5.3.6.

$$\mathrm{BQP}[\mathrm{E_{CF}}] \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$$

Proof.

Let L be a language in BQP[$\mathrm{E_{CF}}$]. Without loss of generality we may assume by Lemma 5.3.3 that L is accepted by a quantum Turing machine having a real-valued transition function. If these values are in $\mathrm{E_{CF}}$ then they satisfy the hypotheses of Lemma 5.3.5. Lemma 5.3.4 tells us that the computation of L is not affected by our choice of the Cauchy functions representing the amplitudes. Let Q be a quantum Turing machine computing L, having amplitudes in E_{CF}, and having polynomial time-bound $q(n)$. Let N be the NTM given by Lemma 5.3.5 associated with Q. Let $h(x)$ be the counting function associated with N. Since N is a polynomial time-bounded NTM, $h \in \#\mathrm{P}$.

From Lemma 5.3.5 $Pr(Q \text{ accepts } x) = tc^{-1}(h(x)) \cdot 2^{-2c \cdot \log(|x|)q(|x|)}$. Thus we can compute $Pr(Q \text{ accepts } x)$ in polynomial time with one query to $h(x)$ and accept x if $Pr(Q \text{ accepts } x)$ exceeds one-half and reject otherwise. Thus, L is in $\mathrm{P}^{\#\mathrm{P}[1]}$. $\square$

CHAPTER 6 CONCLUSIONS AND FUTURE DIRECTIONS

In Chapter 3 we explored the properties of the $\mathrm{BP}_\beta\mathrm{P}$ classes. We determined that for biases β that are rational, p-computable, or BPPcomputable, the class $\mathrm{BP}_\beta\mathrm{P}$ is equal to the class BPP. We also showed that there are β's for which BPP is a proper subset of $\mathrm{BP}_\beta\mathrm{P}$. The β's that we used to show this were potentially noncomputable. It would seem that this amount of complexity in the bias is overkill. We would like to show that there are biases which only slightly more complex than BPP-computable, say probabilistic exponential-time computable, for which the separation holds.

We also explored the relationships between BPP and $\mathrm{BP}_\beta\mathrm{P}$. For real numbers $\beta \neq \gamma$ we would also like to begin exploring the relationship between $\mathrm{BP}_\beta\mathrm{P}$ and $\mathrm{BP}_\gamma\mathrm{P}$. One direction would be to consider this relationship when γ is $\mathrm{BP}_\beta\mathrm{P}$-computable.

In Chapter 3 we dealt only with a single bias. We would also like to begin exploring the "pocketful of change" distributions where one has a finite number of biases to use to create the probability distribution for the witnesses.

Our main result in Chapter 4, the Bias Equivalence Theorem, says that every strongly positive, P-computable, coin-toss probability measure ν is *equivalent* to the uniform probability measure μ, in the sense that

$$\nu_{\mathrm{p}}(\mathcal{C}) = 0 \iff \mu_{\mathrm{p}}(\mathcal{C}) = 0$$

for all classes $\mathcal{C} \in \Gamma$, where Γ is a family that contains P, NP, co-NP, R, BPP, P/Poly, PH and many other classes of interest. It would be illuminating to learn more about which probability measures are, and which probability measures are not, equivalent to μ in this sense.

In Chapter 5 we proposed a definition of BQP based on the computational complexity of the amplitudes associated with the quantum transition function. It is clear that until recently researchers were paying insufficient attention to the complexity of these quantities. We believe that our definition makes this difficulty explicit.

As with probabilistic computation using biased-coin distributions, the ability to inadvertently encode extra information in the quantum parameters means we must be vigilant against this possibility. If we wish to use probabilistic or quantum methods to represent what is 'feasible' then we must make sure that we do not hide excess power in these parameters.

Quantum computation is an area of research whose star is on the rise. We believe that there is a very real possibility of a major paradigm shift in computer science when the problems of implementing quantum methods are overcome. The construction of quantum machines would allow the practical solution of problems now considered beyond our reach with deterministic and even probabilistic techniques.

BIBLIOGRAPHY

[1] Ambos-Spies, Klaus. "Randomness, relativizations and polynomial reducibilities." *Proceedings of the 1st Annual IEEE Conference on Structure in Complexity Theory*. Berkeley, 2-5 June 1986. Los Alamitos CA: IEEE Computer Society Press, 1986. 23–34.

[2] Ambos-Spies, Klaus, Mayordomo, Elvira, and Zheng, Xizhong. "A comparison of weak completeness notions." *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*. Philadelphia, 24-27 May 1996. Los Alamitos CA: IEEE Computer Society Press, 1996. 171–178.

[3] Ambos-Spies, Klaus, Terwijn, Sebastiaan, and Zheng, Xizhong. "Resource bounded randomness and weakly complete problems." *Theoretical Computer Science* , 1996. To appear.

[4] Babai, L. "Trading group theory for randomness." *Proceedings of the 17th Annual ACM Symposium on the Theory of Computation*. Providence, 6-8 May 1985. New York: ACM Press, 1985. 421–429.

[5] Benioff, Paul A. "Review of quantum computation." To appear *Trends in Statistical Physics* by *Council of Scientific Information*, Trivandum, India, 1996.

[6] Bennett, Charles H. "Logical reversibility of computation." *IBM Journal of Research and Development* 17(1973):525–532.

[7] Bennett, Charles H. "Notes on the history of reversible computation." *IBM Journal of Research and Development* 32(1988):16–22.

[8] Bennett, Charles H. "Time/space trade-offs for reversible computation." *SIAM Journal on Computation* 18(1989):766–776.

[9] Bennett, Charles H., Bernstein, Ethan, Brassard, Gilles, and Vazirani, Umesh V. "Strengths and weaknesses of quantum computing." Manuscript, 1994.

[10] Bennett, Charles H. and Gill, John. "Relative to a random oracle A, $\mathrm{P}^A \neq NP^A \neq co-NP^A$ with probability 1. " *SIAM Journal on Computation* 10(1981):96–113.

[11] Berlekamp, E.R. "Factoring polynomials over large finite fields." *Mathematics of Computation* 24(1970):713–735.

[12] Bernstein, Ethan and Vazirani, Umesh V. "Quantum complexity theory." *Proceedings of the 25th Annual ACM Symposium on the Theory of Computation.* San Diego, 16-18 May 1993. New York: ACM Press, 1993. 11–20.

[13] Berthiaume, André. *L'orinateur quantique: complexité et stabilisationdes calculs.* PhD thesis. Université de Montréal, 1995.

[14] Berthiaume, André. "Quantum computation." *Complexity Theory Retrospective II.* Eds. Hemaspaandra, Lane A. and Selman, Alan A. Berlin:Springer-Verlag, to appear.

[15] Berthiaume, André and Brassard, Gilles. "Oracle quantum computing." *Proceedings of the Workshop on Physics and Computation - Physcomp '92.* Dallas, 2-4 October 1992. Los Alamitos, CA: IEEE Computer Society Press, 1992. 195–199.

[16] Berthiaume, André and Brassard, Gilles. "The quantum challenge to structural complexity." *Proceedings of the 7th Annual IEEE Conference on Structure in Complexity Theory*. Boston, 22-25 June 1992. Los Alamitos CA: IEEE Computer Society Press, 1992. 132–137.

[17] Berthiaume, André, Deutsch, David and Jozsa, Richard. "The stabilization of quantum computations." *Proceedings of the Workshop on Physics and Computation - Physcomp '94*. Dallas, 17-20 November 1994. Los Alamitos, CA: IEEE Computer Society Press, 1994. 60–62.

[18] Brassard, Giles. "Cryptology column - quantum computing: The end of classical cryptology?" *SIGACT News* 25(4):15–21.

[19] Brassard, Gilles. "New trends in quantum computing." *Proceedings of the 13th annual Symposium on Theoretical Aspects of Computer Science*. Grenoble, 22-24 February 1996. Lecture Notes in Computer Science 1046. Berlin: Springer-Verlag, 1996. 3–10.

[20] Breutzmann, Josef M. and Lutz, Jack H. "Equivalence of measures for complexity classes." Manuscript, 1996.

[21] Cai, Jin-Yi and Selman, Alan L. "Fine separation of average time complexity classes." *Proceedings of the 13th annual Symposium on Theoretical Aspects of Computer Science*. Grenoble, 22-24 February 1996. Lecture Notes in Computer Science 1046. Berlin: Springer-Verlag, 1996. 331–343.

[22] Chernoff, H. "A measure of asymptotic efficiency for tests of a hypothesis based on a sum of observations." *Annals of Mathematical Statistics* 23(1952):493–507.

[23] Chor, B. and Goldreich, O. "Unbiased bits from sources of weak randomness and probabilistic communication complexity." *SIAM Journal on Computation* 17(1988):230–261.

[24] de Leeuw, K., Moore, E.F., Shannon, C.E., and Shapiro, N. "Computability by probabilistic machines." *Automata Studies* New York:American Mathematical Society, 1956. 183–198.

[25] Deutsch, David. "Quantum theory, the Church-Turing principle and the universal quantum computer." *Proceedings of the Royal Society of London* A400(1985):97–117.

[26] Deutsch, David. "Quantum computational networks." *Proceedings of the Royal Society of London* A425(1989): 73–90.

[27] Deutsch, David and Jozsa, Richard. "Rapid solutions to problems by quantum computation." *Proceedings of the Royal Society of London* A439(1992):553–558.

[28] Feynman, Richard P. "Quantum mechanical computers." *Foundations of Physics* 16(1986):507–531.

[29] Freidzon, R.I. "Families of recursive predicates of measure zero." translated in *Journal of Soviet Mathematics* 6(1976):449–455.

[30] Gill, J. "Probabilistic Turing machines and complexity of computation." PhD thesis. University of California-Berkeley, 1972.

[31] Gill, J. "Computational complexity of probabilistic Turing machines." *SIAM Journal on Computation* 6(1977):675–695.

[32] Hagerup, Torben and Rüb, Christine. "A guided tour of Chernoff bounds." *Information Processing Letters* 33(1989/90):305–308.

[33] Halmos, Paul R. *Measure Theory.* Berlin:Springer-Verlag, 1950.

[34] Hermo, Montserrat. "Nonuniform complexity classes with sub-linear advice functions." PhD thesis. University of the Basque Country, 1996.

[35] Juedes, David W. "Weakly complete problems are not rare." *Computational Complexity* 5(1996):267–283.

[36] Juedes, David L. and Lutz, Jack H. "The complexity and distribution of hard problems." *SIAM Journal on Computation* 24(1995):279–295.

[37] Juedes, David L. and Lutz, Jack H. "Weak completeness in E and E_2." *Theoretical Computer Science* 143(1995):149–158.

[38] Kakutani, S. "On the equivalence of infinite product measures." *Annals of Mathematics* 49(1948):214–224.

[39] Kautz, Steven. Personal communication, 1996.

[40] Ko, Ker-I. "On helping by robust oracle machines." *Theoretical Computer Science* 52(1987):15–36.

[41] Ko, Ker-I. *Complexity Theory of Real Functions.* Boston:Birkhäuser, 1991.

[42] Levine, Robert Y. and Sherman, Alan T. "A note on Bennett's time-space tradeoff for reversible computation." *SIAM Journal on Computation* 19(1990):673–677.

[43] Lutz, Jack H. "Almost everywhere high nonuniform complexity." *Journal of Computer and System Sciences* 44(1992):220–258.

[44] Lutz, Jack H. "The quantitative structure of exponential time." *Proceedings of the 8th Annual IEEE Conference on Structure in Complexity*

Theory. San Diego CA, 18-21 May 1993. Los Alamitos CA: IEEE Computer Society Press, 1993. 158–175. Updated version to appear in L.A. Hemaspaandra and A.L. Selman (eds.), *Complexity Theory Retrospective II*, Springer-Verlag, 1996.

[45] Lutz, Jack H. "Weakly hard problems." *SIAM Journal on Computation* 24(1995):1170–1189.

[46] Lutz, Jack H. "Observations on measure and lowness for Δ_2^{P}." *Proceedings of the 13th annual Symposium on Theoretical Aspects of Computer Science.* Grenoble, 22-24 February 1996. Lecture Notes in Computer Science 1046. Berlin: Springer-Verlag, 1996. 87–97.

[47] Lutz, Jack H. and Mayordomo, Elvira. "Measure, stochasticity, and the density of hard languages." *SIAM Journal on Computation* 23(1994):762–779.

[48] Lutz, Jack H. and Mayordomo, Elvira. "Genericity, measure, and inseparable pairs." In preparation 1996.

[49] Lutz, Jack H. and Mayordomo, Elvira. "Cook versus Karp-Levin: Separating completeness notions if NP is not small." *Theoretical Computer Science* 164(1996):141–163.

[50] Margolus, Norman. "Quantum computation." *Annals of the New York Academy of Sciences* 480(1986):487–497.

[51] Mayordomo, Elvira. "Almost every set in exponential time is P-bi-immune." *Theoretical Computer Science* 136(1994):487–506.

[52] Mehlhorn, Kurt. "The 'almost all' theory of subrecursive degrees is decidable." *Proceedings of the 2nd International Colloquium on Automata, Languages, and Programming.* Berlin:Springer-Verlag, 1974. 317–325.

[53] Nisan, Noam. "Extracting randomness:How and why a survey." *Proceedings of the 11th Annual IEEE Conference on Computational Complexity.* Philadelphia, 24-27 May 1996. Los Alamitos CA: IEEE Computer Society Press, 1996. 171–178.

[54] Oxtoby, J. C. *Measure and Category.* 2nd edition, Berlin:Springer-Verlag 1980.

[55] Papadimitriou, Christos H. *Computational Complexity.* Reading, MA: Addison-Wesley 1994.

[56] Rabin, M.O. "Probabilistic algorithms." *Algorithms and complexity, recent results and new directions.* New York:Academic Press. 21–39

[57] Regan, Kenneth W., Sivakumar, D., and Cai, Jin-Yi. "Pseudorandom generators, measure theory, and natural proofs." *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science.* Milwaukee, 23-25 October 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995. 26–35.

[58] Rogers Jr, H. *Theory of Recursive Functions and Effective Computability.* New York:McGraw - Hill 1967.

[59] Santha, M. and Vazirani, Umesh. "Generating quasi-random sequences from slightly random sources." *Journal of Computer and System Sciences* 33(1986):75–87.

[60] Santos, E. "Probabilistic Turing machines and computability." *Proceedings of the American mathematical Society* 22(1969):704–710.

[61] Schnorr, C. P. "Klassifikation der Zufallsgesetze nach Komplexität und Ordnung." *Z. Wahrscheinlichkeitstheorie verw. Geb.* 16(1970):1–21.

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

[63] Schnorr, C. P. "Zufälligkeit und Wahrscheinlichkeit." *Lecture Notes in Mathematics* number 218, 1971.

[64] Schnorr, C. P. "Process complexity and effective random tests." *Journal of Computer and System Sciences* 7(1973):376–388.

[65] Schöning, Uwe. "Complexity and Structure." *Lecture Notes in Computer Science 21,* Eds. Goos, G. and Hartmanis, J. Berlin: Springer-Verlag 1985.

[66] Schöning, Uwe. "Probabilistic complexity classes and lowness." *Journal of Computer and System Sciences* 39(1989):84–100.

[67] Shor, Peter W. "Algorithms for quantum computation: Discrete log and factoring." *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science.* Santa Fe, 20-22 November 1994. Los Alamitos, CA: IEEE Computer Society Press, 1994. 20–22.

[68] Shor, Peter W. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer." *SIAM Journal on Computation* , 1996, to appear.

[69] Siegelman, Hava T. "On the computational power of probabilistic and faulty neural networks." *Proceedings of the 21st International Colloquium*

on Automata, Languages, and Programming. Jerusalem, 11-14 June 1994. Lecture Notes in Computer Science 820. Berlin: Springer-Verlag, 1994. 23–34.

[70] Simon, David. "On the power of quantum computation." *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science.* Santa Fe, 20-22 November 1994. Los Alamitos, CA: IEEE Computer Society Press, 1994. 116–123.

[71] Sipser, Michael. "A complexity theoretic approach to randomness." *Proceedings of the 15th Annual ACM Symposium on the Theory of Computation.* Boston, 25-27 April 1983. New York: ACM Press, 1983. 330–335.

[72] Solovay, R. and Strassen, V. "A fast Monte-Carlo test for primality." *SIAM Journal on Computation* 6(1977):84–85.

[73] Valiant, Leslie G. "The complexity of computing the permanent." *Theoretical Computer Science* 8(1979):189–201.

[74] van Lambalgen, Michiel. *Random sequences.* PhD thesis. Department of Mathematics, University of Amsterdam, 1987.

[75] Vazirani, Umesh. "Randomness, adversaries and computation." PhD thesis. University of California, Berkeley, 1986.

[76] Vazirani, Umesh and Vazirani, V. "Random polynomial time is equal to slightly-random polynomial time." *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science.* Portland, 21-23 October 1985. Los Alamitos, CA: IEEE Computer Society Press, 1985. 417–428.

[77] von Neumann, John. "Various techniques for use in connection with random digits." in *von Neumann's Collected Works.* New York: Pergamon, 1963. 768–770.

[78] Vovk, V. G. "On a randomness criterion." *Soviet Mathematics Doklady* 35(1987):656–660.

[79] Watrous, John. "On one-dimensional quantum cellular automata." *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science.* Milwaukee, 23-25 October 1995. Los Alamitos, CA: IEEE Computer Society Press, 1995. 528–537.

[80] Yao, Andrew C.-C. "Quantum circuit complexity." *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science.* Palo Alto, 3-5 November 1993. Los Alamitos CA: IEEE Computer Society Press, 1993. 352–361.

[81] Zuckerman, David. "General weak random sources." *Proceedings of the 31th IEEE Symposium on Foundations of Computer Science.* St. Louis, 22-24 October 1990. Los Alamitos, CA: IEEE Computer Society Press, 1990. 534–543.

[82] Zuckerman, David. "Simulating BPP using a general weak random source." *Proceedings of the 32th IEEE Symposium on Foundations of Computer Science.* San Juan, 1-4 November 1991. Los Alamitos, CA: IEEE Computer Society Press, 1991. 79–89.

[83] Zvonkin, A. K. and Levin, Leonid A. "The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms." *Russian Mathematical Surveys* 25(1970):83–124.