To the University of Wyoming:

The members of the Committee approve the dissertation of Adewale Sekoni presented on May 2, 2018.

Dr. John Hitchcock, Chairperson

Dr. Eric Moorhouse, External Department Member

Dr. James Caldwell

Dr. Ruben Gamboa

Dr. Lars Kotthoff

APPROVED:

James Caldwell, Head, Computer Science

Michael Pishko, Dean, College of Engineering and Applied Sciences

Sekoni, Adewale, Polynomial-Time Random Oracles, Nondeterministic Sublinear Time, and Boolean Function Complexity, Ph.D., Computer Science, May, 2018.

We use resource-bounded measure to study nondeterministic sublinear time, random oracles that separate complexity classes, and complexity measures of Boolean functions.

1. **Nondeterministic Sublinear Time.** The measure hypothesis is a quantitative strengthening of the P $\neq$ NP conjecture which asserts that NP is a nonnegligible subset of EXP. Cai, Sivakumar, and Strauss (1997) showed that the analogue of this hypothesis in P is false. In particular, they showed that NTIME$[n^{1/11}]$ has measure 0 in P. We improve on their result to show that the class NTIME$[o(n)]$ of all languages decidable in nondeterministic sublinear time has measure 0 in P. Our result is based on DNF width and holds for all four major notions of measure on P.

2. **Random Oracles that Separate Complexity Classes.** Bennett and Gill (1981) showed that $\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{coNP}^A$ for a random oracle A, with probability 1. We investigate whether this result extends to individual polynomial-time random oracles. We show that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle A that is p-betting-game random. Ideally, we would extend this result to p-random oracles. We show that answering this either way would imply an unrelativized complexity class separation, either P $\neq$ PSPACE or BPP $\neq$ EXP.

   Rossman, Servedio, and Tan (2015) showed that the polynomial-time hierarchy is infinite relative to a random oracle, solving a longstanding open problem. We consider whether we can extend their result to show that PH^A is infinite relative to oracles A that are p-betting-game random. We show that if PH^A separates at even its first level, then the unrelativized separation NP $\neq$ EXP holds.

3. **Complexity Measures of Boolean Functions.** The study of the nonuniform circuit-size complexity of uniform complexity classes using resource-bounded measure was initiated by Lutz (1992). He showed almost all languages in ESPACE have maximum circuit-size complexity. This was subsequently improved to a resource-bounded

dimension result in the third level of the exponential time hierarchy by Hitchcock and Vinodchandran (2006). We use both resource-bounded measure and dimension to study other complexity measures of Boolean functions. The main complexity measures we consider are DNF size, DNF width, DNF-of-parities size, and influence.

POLYNOMIAL-TIME RANDOM ORACLES, NONDETERMINISTIC SUBLINEAR TIME, AND BOOLEAN FUNCTION COMPLEXITY

by

Adewale Sekoni, MS

A dissertation submitted to the
Computer Science
and the
University of Wyoming
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in
COMPUTER SCIENCE

Laramie, Wyoming
May 2018

To Francesca.

Contents

Acknowledgments

I would like to thank my adviser John Hitchcock. You always believed in me even when I didn't believe in myself. Being your student has been an honor and a pleasure, you made this journey fun and challenging. Many thanks to Eric Moorhouse for discussions on topics outside my wheelhouse, you were always welcoming and helpful. I would also like to thank the rest of my committee, Thomas Bailey, James Caldwell, Ruben Gamboa and Lars Kotthoff, your guidance has been invaluable. Thanks to Hadi Shafei for many valuable discussions.

I would also like to thank my mother Adetoun for her love and support, I couldn't have done this without her. To friends and family that helped and supported me in ways, Deborah Ayis, Olutayo Gana, Stephen Nkeonye, Leke Omoteso, Bola Omoteso and Victor Sekoni, thank you all for being there for me.

This research was supported in part by NSF grant 0917417.

Adewale Sekoni

University of Wyoming

May 2018

Chapter 1

Introduction

In this dissertation we study three categories of problems in complexity theory. First we study nondeterministic sublinear time, then random oracles that separate complexity classes, and complexity measures of Boolean functions. The properties of Boolean functions plays a vital role in each of these areas. We use them to study the properties of random languages in various complexity classes. In the following sections we discuss and motivate the reasons for choosing each of these areas.

A central goal of Complexity Theory is to study the limits of computation in the context of limited resources. The most common resources considered are time and space. To achieve this goal various tools and techniques have been developed to answer important questions about effective computation. In this dissertation we use the tools from resource-bounded measure and dimension to study some fundamental properties of problems in various complexity classes.

The complexity classes we consider range from P–the class of problems solvable in polynomial time, to ESPACE = DSPACE($2^{O(n)}$), the class of problems solvable in linear exponential space. It is easy to motivate the study of complexity classes like P since they capture efficient computation. But even classes like E = DTIME($2^{O(n)}$) and EXP = DTIME($2^{n^{O(1)}}$) that capture mostly intractable problems are worth studying for both practical and theoretical reasons. For example, the difficulty of problems in E–the class of problems solvable

in linear exponential time, has implications on the efficient derandomization of efficient randomized algorithms [41]. Furthermore, classes like E are worth studying from a theoretical point of view because the difficulty of problems in smaller complexity classes may have an easier analogue in larger classes. So they provide a good starting point for solving difficult problems. For example, it will be probably easier to separate NP–the class of problems solvable in nondeterministic polynomial time, from EXP than it would be to separate P from NP. Our results also link the nature of the random oracles that separate various levels of the polynomial hierarchy to important open questions in complexity theory. Questions such as whether or not P = PSPACE or BPP = EXP. These are fundamental questions that deal with the importance of space and randomness in computation.

The main tools, techniques and concepts we use are taken from resource-bounded measure [31, 32], resource-bounded dimension [33], circuit complexity theory, and analysis of Boolean functions [44].

Resource-bounded measure (RBM) was introduced by Lutz in [31]. It is generalization of classical Lebesgue measure to complexity classes. It provides a framework that can be used to quantify complexity classes as either big (measure 1), small (measure 0), or immeasurable. We use this theory to give meaning to statements such as "almost all languages in EXP do not have linear size circuits." This is achieved by showing that the class of languages in EXP with linear sized circuits is a measure 0 subset of EXP [31]. RBM uses martingales to characterize measure 0 sets. These measure 0 sets behave like small sets, for example, small unions of measure 0 sets also have measure 0.

An alternative view of RBM is as a "constructive probabilistic method." The probabilistic method [5] is a nonconstructive proof technique in which the existence of objects with certain properties is proved by showing a randomly created object has the desired properties with non-zero probability. Such a proof might provide no clue as to how such objects can be constructed. This is not the case when using RBM. A Resource-bounded random language with property $\mathcal{P}$ can be created within some complexity class depending on the resource bound. The typical resource bounds considered are time and space but it can be applied to

other resources as well. For example, the class of polynomial-time computable martingales gives us a notion of randomness in E. If we show that $\mathcal{P}$ is p-random, then we also know that objects with property $\mathcal{P}$ can be created in E.

Resource-bounded dimension (RBD) [33] is a generalization of classical Hausdorff dimension. With this framework we are able to assign a dimension (some nonnegative number less than or equal to 1) to complexity classes. Both RBM and RBD have been applied to various areas of complexity theory such as circuit complexity theory [12, 25, 31, 33], computational learning theory [17, 20], and structural complexity theory [21, 24]. They have been used to strengthen previously know results and also to prove new ones.

The original formulations of RBM and RBD were used to study complexity classes that contain E. It has since been extended to apply to subexponential complexity classes like P and BPP [2, 3, 39, 49]. Although these extensions generalize the original formulations, they aren't quite as "nice" as the originals. They lose some of the desirable properties of the original formulations. For example, some formulations of measure at P lose the closure of measure 0 sets under finite unions. While some lose the ability to bet on all strings of a given length, without this it is impossible to define dimension in subexponential complexity classes [39]. Fortunately, there is one notion of measure at P due to Moser [39] that allows for defining dimension.

1.1 Nondeterministic Sublinear Time

A central hypothesis of resource-bounded measure [31, 32] is that NP does not have measure 0 in EXP [34, 35]. Cai, Sivakumar, and Strauss [12] proved the surprising result that NTIME[$n^{1/11}$] has measure 0 in P. This implies the analogue of the measure hypothesis in P fails, because NTIME[$\log n$] has measure 0 in P.

We improve the result of Cai et al. by showing that the class of all languages that can be decided in nondeterministic time at most

$$n\left(1 - \frac{2\lg\lg n}{\lg n}\right)$$

has measure 0 in P. In particular, the nondeterministic sublinear time class

$$\mathrm{NTIME}[o(n)]$$

has measure 0 in P.

The result of Cai et al. holds for a notion of measure on P we will refer to as $\Gamma_d(\mathrm{P})$-measure. Moser [39] developed a new notion of measure called F-measure. It is the only notion of measure that allows for defining resource-bounded dimension [33] at P. It was unknown whether or not the result of Cai et al. also holds for F-measure. Our result holds for $\Gamma(\mathrm{P})$ measure (defined in [2]) and therefore for F-measure and all the notions of measure at P considered in [39, 49].

Our stronger result also has a much easier proof than the proof in [12]. Cai et al. use Håstad's switching lemma and pseudorandom generators to show that the class of languages with nearly exponential size circuits has $\Gamma_d(\mathrm{P})$-measure 0 [12]. We use DNF width rather than the circuit size to improve their result. It is well known that a random Boolean function has DNF width close to n (see [14]). In Section 3.1, we show that the class of languages with sublinear DNF width has measure 0 in P. This is then applied in Section 3.2 to show that nondeterministic sublinear time also has measure 0 in P.

1.2 Random Oracles that Separate Complexity Classes

Bennett and Gill [8] initiated the study of random oracles in computational complexity, proving that $\mathrm{P}^A \neq \mathrm{NP}^A$ for a random oracle A, with probability 1. Subsequent work showed that this holds for *individual* random oracles. Book, Lutz, and Wagner [9] showed that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle A that is algorithmically random in the sense of Martin-Löf [37]. Lutz and Schmidt [36] improved this further to show $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle A that is pspace-random [31].

We investigate whether this extends to individual polynomial-time random oracles [6,31]. To show that $\mathrm{P}^A \neq \mathrm{NP}^A$ for p-random oracles A, we need to show that if $\mathrm{P}^A = \mathrm{NP}^A$, then there is a polynomial-time martingale that succeeds on A. This means that if A makes

$\mathrm{P}^A = \mathrm{NP}^A$, then A is somehow predictable or simple.

Allender and Strauss [2] proved that $\{A \mid \mathrm{P}^A \neq \mathrm{BPP}^A\}$ has p-measure 0, which implies that $\mathrm{P}^A = \mathrm{BPP}^A$ for every p-random oracle A. This strengthens another result of Bennett and Gill [8] that $\mathrm{P}^A = \mathrm{BPP}^A$ holds for a random oracle A, with probability 1. Allender and Strauss's proof relies on derandomization [42] and is a different approach than Bennett and Gill. For P vs NP oracles, the best known is the pspace-randomness result of Lutz and Schmidt [36]. In related work, Kautz and Miltersen [30] showed that if A is an algorithmically random oracle, then NP^A does not have p-measure 0. Because the class $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has Hausdorff dimension 1 [19], there is a fundamental limit to how strongly a martingale can succeed on the class.

Each oracle A is associated with a *test language* L_A. This language is tally and $0^n \in L_A$ if and only if in the 2^n tribes of n strings following 0^n, there is at least one tribe contained in A. (See Section 4.1 for a precise definition of L_A. Bennett and Gill used a slightly different, but equivalent formulation of the test language.) It is clear that $L_A \in \mathrm{NP}^A$. From [8], we know that $\{A \mid L_A \in \mathrm{P}^A\}$ has Lebesgue measure 0. Since $\mathrm{P}^A = \mathrm{NP}^A$ implies $L_A \in \mathrm{P}^A$, it follows that $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has measure 0. We would like to show $\{A \mid L_A \in \mathrm{P}^A\}$ has p-measure 0.

Intuitively, if $L_A \in \mathrm{P}^A$, we would like to predict membership of strings in A. This would be relatively simple if the P^A algorithm asked only nonadaptive queries. However, since the queries may be adaptive, there are potentially exponentially many queries–too many to be considered by a polynomial-time martingale.

The difficulty is martingales are forced to bet on strings in lexicographic order. Buhrman et al. [11] introduced an extension of resource-bounded measure using *betting games*. Betting games are similar to martingales but they may adaptively choose the order in which they bet on strings. Whether betting games are equivalent to martingales is an open question [11]. The adaptiveness in betting games allows us to simulate P^A algorithms. We show in Section 4.1 that there is a p-betting game succeeding on $\{A \mid L_A \in \mathrm{P}^A\}$. Therefore $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-betting-game random oracle A.

In Section 4.2, we consider whether there are limitations to extending the betting games result. We show that determining whether or not $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has polynomial-time measure 0 (with respect to martingales) would imply a separation of complexity classes:

- If $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has p-measure 0, then BPP $\neq$ EXP.
- If $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ does not have p-measure 0, then P $\neq$ PSPACE.

This shows that determining the p-measure of $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$, or resolving whether $\mathrm{P}^A \neq \mathrm{NP}^A$ for all p-random A, is likely beyond current techniques.

Bennett and Gill [8] also showed that $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for a random oracle A, with probability 1. Rossman, Servedio, and Tan [46] answered a longtime open question [18] by extending Bennett and Gill's result to separate every level of the polynomial-time hierarchy. They proved an average case depth hierarchy theorem for Boolean circuits which implies that the polynomial-time hierarchy is infinite relative to a random oracle. Can we show that PH is infinite relative to polynomial-time random oracles as well? We show that extending our main result to separate PH^A at even the first level would separate NP from EXP:

- If $\{A \mid \mathrm{NP}^A = \mathrm{coNP}^A\}$ has p-betting-game measure 0, then NP $\neq$ EXP.
- If PH^A is infinite relative to every p-random oracle A, then PH $\neq$ EXP.

1.3 Complexity Measures of Boolean Functions

The study of the nonuniform complexity of languages in uniform complexity classes has been carried out by various researchers. In [31] Lutz strengthened the Shannon circuit-size lower bound [48] of $2^n/n$ from almost every language to almost every language in ESPACE. This result was further improved by Hitchcock et al. [25] using resource-bounded dimension. Cai et al. [12] undertook a similar study, they studied shallow subexponential sized circuits in P. At the heart of these results is the complexity of Boolean functions. We extend these work by considering other complexity measures of Boolean functions.

Boolean functions are fundamental objects in complexity theory. An important property of Boolean functions is how complex they are. There are various complexity measures of Boolean functions such as circuit size, DNF width, and influence. Some of these properties have significant implications on both practical and theoretical computation. For example, the existence of uniformly exponential time computable Boolean functions that require nonuniform exponential size circuits implies the efficient derandomization of polynomial time algorithms [26].

Resource-bounded measure and resource-bounded dimension have been used by various authors to study the circuit and DNF complexity of various complexity classes [12, 31, 47]. We further these areas by considering new complexity measures of Boolean functions. In particular we study DNF size, DNF width, influence, and DNF-of-parity size. We apply these results to obtain new measure results for circuit size and DNF size of languages in various complexity classes.

Given a circuit class $\mathcal{C}$, the circuit size of a Boolean function f with respect to $\mathcal{C}$ is the size of the smallest circuit in $\mathcal{C}$ that computes f. Various circuit classes give rise to different circuit-size complexity measures. We consider three classes of circuits, bounded-depth circuits, DNF circuits and DNF-of-parity circuits.

1. **Bounded-depth circuits** are circuits constructed from AND, OR and NOT gates in which we bound the length from the input gates to the output gate. The bounds we consider are constants and $o(\lg n)$.

2. **DNF circuits** are depth 2 circuits with the first level gates being AND gates and the second level being made up of only one OR gate. The size of a DNF circuit is the fan-in of the single OR gate.

3. **DNF-of-parities circuits** are depth 3 circuits of the from OR $\circ$ AND $\circ$ XOR in which all gates have unbounded fan-in [13]. The bottom level gates are XOR gates whose outputs feed into the next layer of AND gates which in turn feed into the single output OR gate.

We also consider the DNF width and total influence of Boolean functions. These complexity measures are quite different from those previously mentioned but they are related. The width of a DNF circuit is the largest fan-in among its AND gates. The DNF width of a Boolean function is the smallest width of any DNF circuit computing it. The total influence of a Boolean function measures how sensitive a Boolean function is to changes in a single bit of its input, it is also called average sensitivity [43].

We determine for any $\alpha \in [0, 1/2]$ the dimension of languages in E that have influence at most α infinitely often or at most α almost everywhere. We apply this to get new results for the size of shallow depth circuits in E. Like we did with P we once again study the DNF width of languages in E. We apply this is to get new results for the DNF size of language in E. For DNF-of-parities circuits, we strength the result of Cohen and Shinkar. concerning the size of a DNF-of-parity circuit for a random Boolean function [13]. We use this to show that almost all languages in EXP have high DNF-of-parity complexity.

1.4 Organization

1. Chapter 2 covers the preliminaries. Each of the chapters following the preliminaries can be read independently.

2. Chapter 3 covers our results for the measure of nondeterministic sublinear time in P. The results in this chapter also appear in the technical report [22]. This is joint work with John Hitchcock and has been accepted for publication in Theory of Computing Systems (TOCS) pending minor revisions.

3. Chapter 4 covers our results on random oracles that separates P from NP and also various levels of the Polynomial Hierarchy. The results in this chapter also appear in the technical report [23]. This is joint work with John Hitchcock and Hadi Shafei.

4. Chapter 5 covers our results for the nonuniform complexity measures in uniform complexity classes. We study the DNF size, DNF width, Influence and DNF-of-parities

size of languages in complexity classes that contain E. This is unpublished joint work with John Hitchcock.

Chapter 2

Preliminaries

2.1 Languages and Boolean functions

The set of all binary strings is $\{0,1\}^*$. The length of a string $x \in \{0,1\}^*$ is denoted by $|x|$. The empty string is denoted by λ. $\{0,1\}^n$ is the set of strings of length n. $s_0 = \lambda, s_1 = 0, s_2 = 1, s_3 = 00, ...$ is the standard lexicographic enumeration of $\{0,1\}^*$. A language L is a subset of $\{0,1\}^*$. The characteristic sequence of a language L is the sequence $\chi_L \in \{0,1\}^\infty$, where $\chi_L[n] = 1 \iff s_n \in L$. We refer to $\chi_L[s_n] = \chi_L[n]$ as the characteristic bit of s_n in L. A language L can alternatively be seen as a subset of $\{0,1\}^*$, or as an element of $\{0,1\}^\infty$ via identification with its characteristic sequence χ_L. The substring formed by the i^{th} through j^{th} bits of χ_L is denoted by $\chi_L[i,j] = \chi_L[s_i, s_j]$, and $L \upharpoonright s_n$ denotes $\chi_L[0,n]$. Depending on the context $L^{=n}$ represents one of the following:

- $L^{=n} = L \cap \{0,1\}^n$ the set of length n strings of a language L.
- $\chi_L[2^n - 1, 2^{n+1} - 2]$ the substring of the characteristic sequence of L corresponding to the strings in$\{0,1\}^n$.
- A Boolean function to be described in the next section.

Which meaning we are using will be clear from the context.

We denote by $[x, y]$ the set of all strings z such that $x \leq z \leq y$. For any string s_n and number k, $s_n + k$ is the string s_{n+k}; e.g. $\lambda + 4 = 01$. Given $b \in \{0,1\}, x \in \{0,1\}^n$ and $i \in \{1, \cdots, n\}$, $x^{i \mapsto b}$ denotes the string x with its ith bit set to b. $x^{\oplus i}$ denotes x with its ith bit flipped.

We write $\lg n$ for the logarithm to base 2 of n. $(a, b), (a, b], [a, b], [a, b)$ represent intervals of real numbers in the usual way. We say a statement $P(n)$ holds infinitely often (denoted $P(n)$ i.o.) if $P(n)$ is true for infinitely many $n \in \mathbb{N}$. Similarly, we say $P(n)$ holds almost everywhere (denoted $P(n)$ a.e.) if $P(n)$ holds for all but finitely many $n \in \mathbb{N}$.

2.1.1 DNF Formulas and Hamming Subcubes

A Boolean function is any $f : \{0,1\}^n \longrightarrow \{0,1\}$. A DNF (Disjunctive Normal Form) formula of f over the variables $x_1, x_2, \cdots, x_n$ is the logical OR of terms. A term is a logical AND of literals, where a literal is either a variable x_i or its logical negation $\overline{x}_i$. We require that no term contains a variable and its negation [44]. Also the logical OR of the empty term computes the constant **1** function while the the empty DNF computes the constant **0** function. A term's width is the number of literals in it. The size of a DNF computing f is the number of terms in it, while its width is the length of its longest term. The DNF width of f is the shortest width of any DNF computing f. We note that the width of the constant **0** and **1** functions is 0. For any term T we say that T fixes a bit position i if either x_i or its negation appear in T. The bit positions that aren't fixed by T are called free bit positions. For example the term $x_1 x_3 \bar{x}_4 : \{0,1\}^4 \longrightarrow \{0,1\}$, fixes the first, third and fourth bit positions, while the second bit position is free.

We say that term T covers a subset of $\{0,1\}^n$ if it evaluates to true on only the elements of the subset. The subset covered by T is the set of all strings that agree with T on all its fixed bit positions. A string $x \in \{0,1\}^n$ agrees with T if, for any fixed bit position i of T, the ith bit of x is 1 if and only if x_i appears in T. We call the subset covered by T a subcube of dimension $n - k$, where k is the number of literals in T. It is called a subcube because it is a dimension-$(n-k)$ Hamming cube contained in the dimension-n Hamming cube. See

section 2.1.3 for the definition of a Hamming cube.

Associated with any Boolean function is its characteristic string $\chi_f \in \{0,1\}^{2^n}$ defined as

$$\chi_f[w] = 1 \iff f(w) = 1 \text{ for } w \in \{0,1\}^n.$$

For any language L we view $L^{=n}$ as the Boolean function $\chi_{L^{=n}}$ defined as

$$\chi_{L^{=n}}(w) = 1 \iff \chi_L[w] = 1 \text{ for all } w \in \{0,1\}^n.$$

2.1.2 Random Boolean Functions

We set up a few more notations from [45] for discussing random Boolean Functions. The DNF size of a Boolean function f is the denoted by $\ell(f)$, it is the size of the smallest DNF computing f. $\overline{\ell}(n)$ is the expectation of $\ell(f)$, the expectation is taken over the uniform distribution on $f : \{0,1\}^n \longrightarrow \{0,1\}$ i.e. $\overline{\ell}(n)$ is the average DNF complexity of a random Boolean function on n variables. The DNF complexity of a language L is the function $D_L : \mathbb{N} \longrightarrow \mathbb{N}$ defined by $D_L(n) = \ell(L^{=n})$ i.e. $D_L(n)$ is the size of the smallest DNF computing the function whose characteristic string is $\chi_{L^{=n}}$.

2.1.3 Hamming cubes and Influence

The n-dimensional Hamming cube, denoted B_n is the graph with vertex set $\{0,1\}^n$ and an edge between any set of vertices that differ in only one bit position. A subcube of B_n is a subgraph of B_n that is also a hamming cube i.e. the subgraph is isomorphic to a Hamming cube. An edge $\{x, y\}$ of B_n is called a dimension-i edge if x and y differ only in their ith bit position. It is easy to see that B_n has $n2^{n-1}$ edges and they can be partitioned into n sets

$$E_{n,i} = \{\{x, y\} | \{x, y\} \text{ is a dimension-}i \text{ edge of } B_n\},$$

with each set consisting of all dimension-i edges.

Given $f : \{0,1\}^n \longrightarrow \{0,1\}$, B_n^f denotes the two coloring of B_n by f in which every vertex x is colored $f(x)$. An edge $\{x, y\}$ of B_n^f is called a boundary edge if $f(x) \neq f(y)$.

Associated with any dimension-k subcube $B_{n,k}$ of B_n are sets $S_{\text{fix}}, S_{\text{free}} \subseteq [n]$ such that $S_{\text{fix}} = [n] \setminus S_{\text{free}}$ and $|S_{\text{free}}| = k$. All vertices of $B_{n,k}$ have the same bits at all bit positions indexed by S_{fix}. We refer to the numbers in S_{fix} as the fixed bit positions of $B_{n,k}$, while those in S_{free} are referred to as the free bit positions. Thus, when we specify all the fixed bits, we would also have uniquely identified a dimension-k subcube of B_n. It is easy to see that there are $2^{(n-k)}\binom{n}{k}$ dimension-k subcubes of B_n.

For $f : \{0,1\}^n \longrightarrow \{0,1\}$, $I_i[f]$ is the influence of f at coordinate $i \in [n]$. $I_i[f]$ is the probability that the value of $f : \{0,1\}^n \longrightarrow \{0,1\}$ changes when $x \in \{0,1\}^n$ is uniformly selected and the ith bit of x is flipped. $I[f]$ is the total influence of f, it is the sum of the influences of each coordinate of f i.e. $I[f] = \sum_{i=1}^{n} I_i[f]$.

2.1.4 DNF of Parities Circuits

A DNF-of-parity circuit of a Boolean function is a depth 3 circuit of the from OR $\circ$ AND $\circ$ XOR in which all gates have unbounded fan-in [13]. It is a DNF formula applied the parities of the input variables. The size of a DNF-of-parity circuit is the fan-in of the top OR gate. This is analogous to the size of a DNF formula which is also the fan-in of the OR gate. We denote by $DNF_{\oplus}(f)$, the size of the smallest smallest DNF-of-parity circuit computing the Boolean function f. From a combinatorial point of view $DNF_{\oplus}(f)$ is the smallest number of affine subspaces required to cover $f^{-1}(1)$ while the DNF size of f if the number of Hamming cubes required to cover $f^{-1}(1)$. Since a Hamming cube is also an affine space DNF-of-parity circuits are a generalization of a DNF formulas.

2.2 Martingales and Betting Games

We now give a brief overview of martingales and betting games, and how they are applied in computational complexity to define resource-bounded measures and randomness notions. For further details, we refer to [6, 11, 17, 31, 32].

Betting games, which are also called nonmonotonic martingales, originated in the field

of algorithmic information theory. In that setting they yield the notion of Kolmogorov-Loveland randomness (generalizing Kolmogorov-Loveland stochasticity) [38,40]. The concept was introduced to computational complexity by Buhrman et al. [11]. First, we recall the definition of a martingale:

Definition. A *martingale* is a function $d : \{0,1\}^* \to [0,\infty)$ such that for all $w \in \{0,1\}^*$*, we have the following averaging condition:

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

Intuitively, a martingale is betting in order on the characteristic sequence of an unknown language. The martingale starts with finite initial capital $d(\lambda)$. The quantity $d(w)$ represents the current capital the martingale has after betting on the first $|w|$ bits of a sequence that begins with w. The quantities $\pi(w,0) = d(w0)/2d(w)$ and $\pi(w,1) = d(w1)/2d(w)$ represent the fraction of its current capital that the martingale is wagering on 0 and 1, respectively, being the next bit of the sequence. This next bit is revealed and the martingale has $d(w0) = 2\pi(w,0)d(w)$ in the case of a 0 and $d(w1) = 2\pi(w,1)d(w)$ in the case of a 1.

Betting games are a generalization of martingales and have the additional capability of selecting which position in a sequence, or equivalently, which string in a language, to bet upon next. A betting game is permitted to select strings in a nonmonotone order, that is, it may bet on longer strings, then shorter strings, then longer strings again (with the important restriction that it may not bet on the same string twice). Like martingales, betting games must also satisfy the averaging law, i.e. the average of the betting game's capital after betting on a string s when s belongs and when s doesn't belong to the language is the same as its capital before betting on s. We use the following definition of a betting game from [11].

Definition. A betting game G is an oracle Turing machine that maintains a "capital tape" and a "bet tape," in addition to its standard query tape and worktapes. The game works in rounds $i = 1, 2, 3, \ldots$ as follows. At the beginning of each round i, the capital tape holds a nonnegative rational number C_{i-1}. The initial capital C_0 is some positive rational number. G computes a query string x_i to bet on, a bet amount $B_i, 0 \leq B_i \leq C_{i-1}$, and

a bet sign $b_i \in \{-1, +1\}$. The computation is legal so long as x_i does not belong to the set $\{x_1, \cdots, x_{i-1}\}$ of strings queried in earlier rounds. G ends round i by entering a special query state. For a given oracle language A, if $x_i \in A$ and $b_i = +1$, or if $x_i \notin A$ and $b_i = -1$, then the new capital is given by $C_i := C_{i-1} + B_i$, else by $C_i := C_{i-1} - B_i$. We charge M for the time required to write the numerator and denominator of the new capital C_i down. The query and bet tapes are blanked, and G proceeds to round $i + 1$.

It is easy to see from the above definition that b_i and B_i can easily be computed from the current capital $C_i := C_{i-1} + b_i B_i$ of the betting game. Therefore, we can equivalently define a betting game by describing the computation of the current capital C_i without explicitly specifying the computation of b_i and B_i. We do this because it is clearer and more intuitive to describe the computation of the current capital of the betting game presented in Chapter 4.

Definition. If a betting game G earns unbounded capital on a language A (in the sense that for every constant c there is a point at which the capital exceeds c when betting on A), we say that G *succeeds on* A. The *success set* of a betting game G, denoted $S^\infty[G]$, is the set of all languages on which G succeeds. A betting game G *succeeds on* a class X of languages if $X \subseteq S^\infty[G]$.

By adding a resource bound Δ on the computation of a betting game or martingale, we get notions of resource-bounded measure on $\{0,1\}^\infty$. For Chapter 4 and 5 the resource bounds we use are p = DTIMEF($n^{O(1)}$), p_2 = DTIMEF($2^{(\lg n)^{O(1)}}$), and pspace = DSPACEF($n^{O(1)}$). We say a class $X \subseteq \{0,1\}^\infty$ has Δ-betting-game measure 0, if there is a Δ-computable betting game that succeeds on every language in it. It has Δ-measure 0 if the betting game is also a martingale [31]. A class X has Δ-betting-game measure 1 if X^c has Δ-betting-game measure 0. Similarly, X has Δ-measure 1 if X^c has Δ-measure 0. A language A is Δ-betting-game random if there is no Δ-computable betting game that succeeds on A. Similarly, A is Δ-random if there is no Δ-computable martingale that succeeds on A.

The ability of the betting game to examine a sequence nonmonotonically makes determining its running time complicated, since each language can induce a unique computation

of the betting game. In other words, the betting game may choose to examine strings in different orders depending upon the language it is wagering against. Buhrman et al. looked at a betting game as an infinite process on a language, rather than a finite process on a string. They used the following definition:

Definition. A betting game G runs in time $t(2^n)$ if for all languages A, every query of length n made by G occurs in the first $t(2^n)$ steps of the computation.

Specifically, once a $t(2^n)$-time-bounded betting game uses $t(2^n)$ computational steps, it cannot go back and select any string of length n. We only consider a polynomial time betting game in Chapter 4. Most importantly, no polynomial-time betting game can succeed on the class EXP = DTIME($2^{n^{O(1)}}$).

2.2.1 Martingales and Betting Games: Intuitive view

Intuitively, a betting game can be viewed as the strategy of a gambler who bets on infinite sequence of strings. The gambler starts with initial capital C, then begins to query strings to bet on. The gambler's goal is to grow the capital C without bound. The same view holds for martingales with the restriction that the gambler must bet on the strings in the standard ordering.

Lemma 2.2.1 (Δ-Ideal Lemma [31])**.** *Let $\mathcal{F}$ be either the collection $\mathcal{F}_\Delta$ of all Δ-measure 0 sets or the collection $\mathcal{F}_{R(\Delta)}$ of all sets that have measure 0 in $R(\Delta)$. In either case, $\mathcal{F}$ has the following three closure properties.*

1. *If $X \subseteq Y \in \mathcal{F}$, then $X \in \mathcal{F}$.*
2. *If X is a finite union of elements of $\mathcal{F}$, then $X \in \mathcal{F}$.*

2.3 Resource-Bounded Measure at P

An apparently more difficult task is developing a notion of resource-bounded measure on subexponential classes, in particular developing a measure on P [4]. There are at least four

notions of measure defined on P. Three of these are discussed by Strauss [49] and the other is discussed by Moser [39]. None of them are quite as "nice" as measures on complexity class $C \supseteq \mathrm{E}$, each one of them having some limitations. See [4, 39, 49] for a more detailed discussion of the limitations of these notions of measure. In this paper we only consider one notion of measure on P we call $\Gamma(\mathrm{P})$-measure. $\Gamma(\mathrm{P})$-measure was introduced in [3]. We use $\Gamma(\mathrm{P})$-measure for two reasons. First, it is the simplest of the four notions of measure on P. Second, the martingales considered in $\Gamma(\mathrm{P})$-measure can be easily shown to be martingales in the other notions of measure at P [39, 49].

2.3.1 $\Gamma(\mathrm{P})$-measure

A $\Gamma(\mathrm{P})$-martingale is a martingale d such that:

- $d(w)$ can be computed by a Turing machine M with oracle access to w and input $s_{|w|}$. We denote this computation as $M^w(s_{|w|})$.
- $M^w(s_{|w|})$ is computed in time polynomial in $\lg(|w|)$. In other words, the computation is polynomial in the length of $s_{|w|}$.
- d only bets on strings in a P-printable set denoted G_d.

The input string $s_{|w|}$ to $M^w(s_{|w|})$ allows the Turing machine to compute the length of w without reading all of w whose length is exponential in the length of $s_{|w|}$. A set $S \subseteq \{0,1\}^*$ is P-printable [1] if $S \cap \{0,1\}^n$ can be printed in time polynomial in n. A class $C \subseteq \{0,1\}^\infty$ has $\Gamma(\mathrm{P})$-measure 0 zero if there is some $\Gamma(\mathrm{P})$-martingale that succeeds on it [49].

2.4 Gales and Resource-Bounded Measure and Dimension

In this section we consider gales, a generalization of martingales. The follow definitions and notation are taken from [33]. For each $i \in \mathbb{N}$ we define a class G_i of functions on $\mathbb{N}$ as follows:

$$G_0 = \{g | \exists k \text{ such that, } f(n) \leq kn, \text{ for all but finitely many } \mathbb{N}\},$$
$$G_{i+1} = 2^{G_i(\lg n)} = \{g | \exists h \in G_i \text{ such that, } f(n) \leq 2^{h(\lg n)}, \text{ for all but finitely many } \mathbb{N}\}.$$

The functions in the G_is are regarded as growth functions. We use them to define the complexity classes, $\mathrm{E}_i = \mathrm{DTIME}(2^{G_{i-1}})$ and $\mathrm{E}_i\mathrm{SPACE} = \mathrm{DSPACE}(2^{G_{i-1}})$ for $i \geq 1$. For $i \geq 1$ we also define the following classes of functions on $\{0,1\}^*$:

$$\mathrm{p}_i = \{g : \{0,1\}^* \longrightarrow \{0,1\}^* | g \text{ is computable in } G_i \text{ time.}\}$$
$$\mathrm{p}_i\mathrm{space} = \{g : \{0,1\}^* \longrightarrow \{0,1\}^* | g \text{ is computable in } G_i \text{ space.}\},$$

where the length of the output $g(x)$ is included as part of the space required to compute g. We write p for p_1 and pspace for $\mathrm{p}_1$space. In this dissertation Δ will be one of the classes p_i, or $\mathrm{p}_i$space for $i \geq 1$.

A constructor is a function $\delta : \{0,1\}^* \longrightarrow \{0,1\}^*$ such that x is a proper prefix of $\delta(x)$. The unique language constructed by δ is the language $R(\delta) \in \{0,1\}^\infty$ such that, $\delta^n(\lambda)$ is a prefix of $R(\delta)$ for all $n \in \mathbb{N}$. We write $R(\Delta)$ for the set of languages $R(\delta)$ such that, δ is a constructor that can be computed in Δ. It is easy to see that for any $i \geq 1$, $R(\mathrm{p}_i) = \mathrm{E}_i$, and $R(\mathrm{p}_i\mathrm{space}) = \mathrm{E}_i\mathrm{SPACE}$.

Definition. We say X has measure 0 in $R(\Delta)$, and we write $\mu(X|R(\Delta)) = 0$, if $\mu_\Delta(X \cap R(\Delta)) = 0$. A set has measure 1 in $R(\Delta)$, and we write $\mu(X|R(\Delta)) = 1$, if $\mu_\Delta(X^c \cap R(\Delta)) = 0$. If $\mu(X|R(\Delta)) = 1$ we say that almost every language in $R(\Delta)$ is in X.

Theorem 2.4.1 (Lutz [31])**.** $\mu(R(\Delta)|R(\Delta)) \neq 0$

2.4.1 Gales

We now introduce gales which are a generalization of martingales, they are used to define resource-bounded dimension. The following definitions are taken from [33].

Definition. Let $s \in [0, \infty)$.

1. An s-supergale is a function $d : \{0,1\}^* \longrightarrow [0, \infty)$ that satisfies the condition

$$d(w) \geq 2^{-s}(d(w0) + d(w1)) \tag{2.4.1}$$

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

2. An s-gale is an s-supergale that satisfies 2.4.1 with equality for all $w \in \{0,1\}^*$.

3. A martingale is a 1-gale while a supermartingale is a 1-supergale.

Definition. Let d be an s-supergale, where $s \in [0, \infty)$.

1. We say a language d succeeds on a language $A \in \{0,1\}^\infty$ if

$$\limsup_{n \to \infty} d(A[0, n-1]) = \infty.$$

2. The success set of d is

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

3. We say d succeeds on $X \subseteq \{0,1\}^\infty$ if X is contained in the success set of d.

Just like martingales and betting games we can put a resource bound Δ on the computation of s-gales. This yields the notion of resource-bounded dimension. Intuitively, the s parameter in the definition of an s-gale can bee seen as a tax rate parameter. The lower the value of s the higher the tax paid by the s-gale. Therefore, whenever $s < s'$, it is more difficult to design an s-gale than an s'-gale that succeeds on a given set.

Definition. Let $X \subseteq \{0,1\}^\infty$, and $\mathcal{G}_\Delta(X)$ be the set of all $s \in [0, \infty)$ such that there is a Δ-computable s-gale d for which $X \subseteq S^\infty[d]$.

1. The Δ-dimension of X is $\dim_\Delta(X) = \inf \mathcal{G}_\Delta(X)$.

2. The dimension of X in $R(\Delta)$ is $\dim(X|R(\Delta)) = \dim_\Delta(X \cap \mathrm{E})$.

Informally, the dimension of X is maximum tax rate (minimum s) for which there exists an s-gale that succeeds on X.

Observation 2.4.2. *If $X \subseteq \{0,1\}^\infty$ has Δ-dimension (dimension in $R(\Delta)$) less than 1, then X has Δ-measure 0 (measure 0 in $R(\Delta)$).*

Theorem 2.4.3 (Lutz [33])**.** $\dim(R(\Delta)|R(\Delta)) = \dim_\Delta(R(\Delta)) = 1$

Chapter 3

Nondeterministic Sublinear Time

In this section we present an improvement of the result of Cai, Sivakumar and Strauss [12]. They showed that the class of all languages decidable in nondeterministic time at most $n^{1/11}$ has measure 0 in P. Thereby showing that the analogue of the Lutz's measure hypothesis [34, 35] doesn't hold in P. We extend their result in two directions. First, we improve the nondeterministic time bound to $n\left(1-\frac{2\lg\lg n}{\lg n}\right)$. Second, we show that this result holds in two additional notions of measure at P. Therefore showing that the analogue of the measure hypothesis fails in all notions of measure at P.

3.1 Measure and DNF Width at P

In this section we show that the class of languages with sublinear DNF width has measure 0 in P. Given a language L let $\mathrm{DNF}_{\mathrm{width}}(L^{=n})$ denote the DNF width of the Boolean function $L^{=n}$.

Theorem 3.1.1. *The class*

$$X=\left\{L\in\{0,1\}^{\infty}\;\middle|\;\mathrm{DNF}_{\mathrm{width}}(L^{=n})\leq n\left(1-\frac{2\lg\lg n}{\lg n}\right)\text{ i.o.}\right\}$$

has $\Gamma(\mathrm{P})$*-measure 0.*

Proof. For clarity we omit floor and ceiling functions.

The Martingale

Consider the following martingale d that starts with initial capital 4. Let L be the language d is betting on. d splits its initial capital capital into portions $C_{i,1}, C_{i,2}, i \in \mathbb{N}$, where $C_{i,1} = C_{i,2} = 1/i^2$. $C_{n,1}$ and $C_{n,2}$ are reserved for betting on strings in $\{0,1\}^n$. For each length n, d only risks $C_{n,1}$ and $C_{n,2}$. Thus, d never runs out of capital to bet on $\{0,1\}^n$ for all $n \in \mathbb{N}$.

Now we describe how d bets on $\{0,1\}^n$ with $C_{n,1}$. d uses $C_{n,1}$ to bet that the first n strings of $\{0,1\}^n$ don't belong to L. If d makes no mistake then the capital $C_{n,1}$ grows from $1/n^2$ to $2^n/n^2$. But once d makes a mistake it loses all of $C_{n,1}$, i.e. $C_{n,1}$ becomes 0.

Next we describe how the martingale d bets on $\{0,1\}^n$ with $C_{n,2}$. The martingale d only bets with capital $C_{n,2}$ if it loses $C_{n,1}$, i.e. d makes a mistake on the first string of length n that belongs to L. Let us call this string w. We will use w to determine how d bets with $C_{n,2}$. Let $w_1, w_2, \cdots, w_{n/\lg n}$ be a partition of w into $n/\lg n$ substrings, such that $w = w_1 w_2 \cdots w_{n/\lg n}$, and each w_i has length $\lg n$. Each substring w_i specifies a subset $\mathcal{S}_{w_i}$ of dimension $2\lg\lg n$ subcubes that contain w. $\mathcal{S}_{w_i}$ consists of exactly those dimension $2\lg\lg n$ subcubes that contain w, and whose free bit positions are in the range $[(i-1)(\lg n)+1, i\lg n]$. In other words, $\mathcal{S}_{w_i}$ is the set of subcubes that contain w, and have their free bit positions consist entirely of the bit positions of w that were used to form w_i. We will refer to the subcubes in $\mathcal{S}_{w_i}$ as the boundary subcubes of w. It is easy to see that there are $\binom{\lg n}{2\lg\lg n}\frac{n}{\lg n} = n^{1+o(1)}$ boundary subcubes of w. Each boundary subcube will be used to bet on the membership of some strings in $\{0,1\}^n$. d splits $C_{n,2}$ into $\binom{\lg n}{2\lg\lg n}\frac{n}{\lg n}$ equal parts $C_{n,2,i}$, for $i \in [1, \binom{\lg n}{2\lg\lg n}]$. Each part will be used by a boundary subcube for betting.

Finally, to completely specify d, we describe how it bets with each $C_{n,2,i}$ on any string $x \in \{0,1\}^n$ that comes after w, the string d lost all of $C_{n,1}$ on. d bets as follows:

Intuitively, each $C_{n,2,i}$ is reserved for betting on a boundary subcube of w. The martingale predicts that each boundary subcube is contained in $L^{=n}$. If the subcube B_i which contains w is really contained in $L^{=n}$, then the capital reserved for betting on this subcube grows from $C_{n,2,i}$ to $2^{2^{2\lg\lg n}-1}C_{n,2,i}$. This follows because the martingale doesn't make any

```
for each boundary subcube B_i of w do
    C_{n,2,i} ← current capital reserved for betting on B_i
    if x ∈ B_i then
        bet all of C_{n,2,i}on x being in L
    end if
end for
```

mistakes while betting on the $2^{2\lg\lg n} - 1$ strings in $B_i \setminus \{w\}$, and each of these bets doubles $C_{n,2,i}$. Otherwise, if B_i is not contained in $L^{=n}$ then the martingale will make a wrong prediction and lose all its capital reserved for betting on B_i.

The Martingale's Winnings on X

We now show that d succeeds on any $L \in X$ by examining its winnings on $L^{=n}$.

In the first case, suppose the first n strings of $\{0,1\}^n$ are all not contained in L. In this case we bet with $C_{n,1}$ and raise this capital from $1/n^2$ to $2^n/n^2$.

In the second case, suppose $\mathrm{DNF}_{\mathrm{width}}(L^{=n}) \leq n(1 - \frac{2\lg\lg n}{\lg n})$ and one of the first n strings of $\{0,1\}^n$ is in L. Let us denote the first such string by w. In this case d will lose all of $C_{n,1}$ and have to bet with $C_{n,2}$. Since $\mathrm{DNF}_{\mathrm{width}}(L^{=n}) \leq n(1 - \frac{2\lg\lg n}{\lg n})$, w must be contained in a subcube of dimension at least $(\frac{2\lg\lg n}{\lg n})n$ i.e. w is contained in subcube with at least $(\frac{2\lg\lg n}{\lg n})n$ free bit positions. Since $w = w_1 w_2 \cdots w_{n/\lg n}$, one of the w_is must have $2\lg\lg n$ free bit positions. Thus, there must be at least one boundary subcube of w that is contained in $L^{=n}$. Since d must bet on such a subcube, its capital reserved for this subcube rises from $C_{n,2,i} = n^{1+o(1)}$ to $2^{2^{2\lg\lg n}-1} C_{n,2,i} = \Theta(n^{\lg n})$.

Since any $L \in X$ satisfies the above two cases infinitely often, d's capital rises by $\Omega(n^{\lg n})$ infinitely often. Thus, d succeeds on X.

The Martingale is a $\Gamma(\mathrm{P})$-Martingale

Now we need to show d is a $\Gamma(\mathrm{P})$-martingale. It is easy to see that d is computable in time polynomial in n. Since for each $x \in \{0,1\}^n$ we bet on, we iterate though $n^{1+o(1)}$ subcubes of dimension $2\lg\lg n$, and each subcube contains $O(\lg^2 n)$ points. Also the set of strings that

d bets on in $\{0,1\}^n$ is P-printable since it only bets on the $n^{2+o(1)}$ points in the boundary subcubes of the first n strings of length n. □

3.2 Measure and Nondeterministic Time

The following lemma is a generalization of an observation made in [12].

Lemma 3.2.1. *For all n, if $L^{=n}$ can be decided by a nondeterministic Turing machine in time $f(n) \leq n$, then $L^{=n}$ has DNF width at most $f(n)$.*

Proof. If $L^{=n} = \emptyset$, then it is covered by the empty DNF which has width 0. All that's left is to show that $L^{=n}$ is covered by subcubes of dimension at least $n - f(n)$ whenever $L^{=n} \neq \emptyset$. This is sufficient because every subcube of dimension at least $n - f(n)$ is covered by a width $f(n)$ term, so L can be covered by a width $f(n)$ DNF. Let M be a nondeterministic Turing machine that decides L in time at most $f(n)$ and $x \in L^{=n}$. Thus, there is a nondeterministic computation of M on input x that accepts. Since M uses at most $f(n)$ time it can only examine at most $f(n)$ bits of x. So there are at least $n - f(n)$ bits of x that aren't examined by M on some accepting computation of M on x. Therefore, the set of all strings $y \in \{0,1\}^n$ that agree with x in all the bit positions examined by an accepting computation must also be accepted by the same computation. This set of strings is precisely a subcube of dimension at least $n - f(n)$; therefore, it is covered by a DNF term of width at most $f(n)$. Since $x \in L^{=n}$ was arbitrary, it follow that $L^{=n}$ can be covered by DNF term(s) of width at most $f(n)$; therefore, $L^{=n}$ has DNF width at most $f(n)$. □

Theorem 3.2.2. *The class of all languages decidable in nondeterministic time at most $n(1 - \frac{2 \lg \lg n}{\lg n})$ infinitely often has $\Gamma(\mathrm{P})$-measure 0.*

Proof. By Lemma 3.2.1, any language decidable in nondeterministic time at most $n(1 - \frac{2 \lg \lg n}{\lg n})$ has DNF width at most $n(1 - \frac{2 \lg \lg n}{\lg n})$ for all but finitely many n. Therefore, it follows by Theorem 3.1.1 that the set of all such languages have $\Gamma(\mathrm{P})$-measure 0. □

We now have the main result of the paper:

Corollary 3.2.3. NTIME $\left[n\left(1-\frac{2\lg\lg n}{\lg n}\right)\right]$ *has* $\Gamma(\mathrm{P})$*-measure 0.*

Corollary 3.2.4. NTIME$[o(n)]$ *has* $\Gamma(\mathrm{P})$*-measure 0.*

Because $\Gamma(\mathrm{P})$ measure 0 implies measure 0 in the other notions of measure on P [39,49], Theorem 3.2.2 and its corollaries extend to these measures as well.

Corollary 3.2.5. *The class of all languages decidable in nondeterministic time at most* $n(1-\frac{2\lg\lg n}{\lg n})$ *infinitely often has F-measure 0,* $\Gamma_d(\mathrm{P})$*-measure 0, and* $\Gamma/(\mathrm{P})$*-measure 0.*

A language L has decision tree depth $f(n):\mathbb{N}\longrightarrow\mathbb{N}$ infinitely often if $\chi_{L^{=n}}$ has decision tree depth at most $f(n)$ for infinitely many n. It is easy to show and well known that a function with decision tree depth k has DNF width at most k. See [44] for the definition of decision tree depth and a proof of the previous statement. Therefore, Theorem 3.2.2 immediately implies the following corollary.

Corollary 3.2.6. *The set of all languages with decision tree depth at most* $n(1-\frac{2\lg\lg n}{\lg n})$ *infinitely often has* $\Gamma(\mathrm{P})$*-measure 0.*

3.3 Conclusion

DNF width is the main tool used for the proofs of this section. In particular, it works well with $\Gamma(\mathrm{P})$-martingales. In Chapter 5 we consider other complexity measures of Boolean functions. These complexity measures are apparently much harder to work with for the current notions of measures at P. For example, the result of Cai et al. could also have been improved using the influence of a Boolean function. But we were unable to design a martingale for any of the notions of measure at P that succeeds on languages with low influence. For future work we are considering extending the results presented to other sublinear time complexity classes.

Chapter 4

Random Oracles for P vs. NP

In this section we study the relativized P vs. NP problem. Bennett and Gill showed that $\mathrm{P}^R \neq \mathrm{NP}^R$ with probability 1 for a randomly selected oracle R [8]. This result was strengthened to pspace-randomness by Lutz and Schmidt [36]. In related work, Kautz and Miltersen [30] showed that if A is an algorithmically random oracle, then NP^A does not have p-measure 0. We continue on this line of research by showing that the set of all oracles R such that $\mathrm{P}^R = \mathrm{NP}^R$ has p-betting-game measure 0. This implies that $\mathrm{P}^R \neq \mathrm{NP}^R$ with probability 1 for p-betting-game random oracles. Finally, we present limitations to extending our results to other notions of randomness or the polynomial hierarchy.

4.1 Betting Game Random Oracles

In this section we show that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-betting-game random oracle.

Theorem 4.1.1. *The class* $\{A \mid \mathrm{P}^A \neq \mathrm{NP}^A\}$ *has* p*-betting-game measure 1. In particular,* $\mathrm{P}^A \neq \mathrm{NP}^A$ *for every* p*-betting-game random oracle* A*.*

Proof. Given a language A we define the test language

$$L_A = \{0^n \mid \mathrm{Tribes}_{2^n,n}(A[0^n + 1, 0^n + n2^n]) = 1\},$$

where $\mathrm{Tribes}_{2^n,n} : \{0,1\}^{n2^n} \longrightarrow \{0,1\}$ is defined as follows. Given $w \in \{0,1\}^{n2^n}$, first we view w as concatenation of 2^n length n strings $w_1, w_2, \cdots, w_{2^n}$; i.e. $w = w_1 w_2 \cdots w_{2^n}$.

$\text{Tribes}_{2^n,n}(w)$ is 1 if and only if $w_i = 1^n$ for some i. Secondly, we view w as the substring $A[0^n + 1, 0^n + n2^n]$ of the characteristic sequence of some language A. With both views in mind, we define a tribe to be the set of strings whose characteristic bits are encoded by some w_i. For example, given any $i \in [1, 2^n]$, the set of strings $[0^n + (i-1)n + 1, 0^n + in]$ is a tribe because its characteristic bits are encoded by w_i. Since the n strings in any tribe have length $O(n)$, an NP oracle machine can easily verify the membership of any 0^n, therefore $L_A \in \text{NP}^A$. Now we define a betting game G that succeeds on the set $X = \{A \mid \text{P}^A = \text{NP}^A\}$, thereby proving the theorem. Our betting game G is going to simulate oracle Turing machines on some strings in the set $\{0^n \mid n \in \mathbb{N}\}$. Let $M_1, M_2, \cdots$ be an enumeration of all oracle TMs, where M_i runs in time at most $n^{\lg i} + i$ on inputs of length n. The initial capital of G is 2 and we view it as composed of infinite "shares" $a_i = b_i = 2^{-i}, i \in \mathbb{N}$ that are used by G to bet on some of the strings it queries.

Before we go into the details of the implementation of G, we give a high level view. The strategy of G to succeed on X is quite simple. For any language A, the cardinality of $\{0^n \mid 0^n \notin L_A\}$ is either finite or infinite. When it is finite, after seeing a finite number of strings all following strings will belong to L_A. G uses "shares" a_i reserved at its initialization to bet in this situation. On the other hand when it is infinite and $A \in X$ we can find an oracle TM M_i that decides L_A. Most importantly this TM rejects its input infinitely often and it is only in this situation that we bet with the b_i "shares". Details follow.

First we specify the order in which G queries strings followed by which strings it bets on. G operates sequentially in stages $1, 2, \cdots$. In stage j, G queries 0^{n_j}, where n_j is the smallest integer such that all the strings queried in stage $j-1$ have length less than n_j. G then runs the oracle TM M_{i+1} on 0^{n_j}, where i is the number of TMs simulated in the previous stages whose output was inconsistent with L_A in one of the previous stages. During the simulation of M_{i+1}, G answers any queries made by the TM either by looking up the string from its history, or if the string isn't in its history, then G queries it. After the simulation, G queries in the standard lexicographic order all the strings in the 2^{n_j} tribes that follow 0^{n_j} that haven't already been queried. Finally, to complete stage j, G queries all the remaining

strings of length at most the length of the longest string queried by G so far.

Now we specify which strings G bets on and how it bets with the a_i's and b_i's. In stage j, let i and n_j be such that M_i is the Turing machine simulated in this stage and 0^{n_j} is the input it will be simulated on. The only strings G bets on will be the $n_j 2^{n_j}$ strings following 0^{n_j}; i.e. the tribes. We use a_l and b_i, two of the infinite "shares" of our initial capital reserved by G for betting, where l is the smallest positive integer such that $a_l \neq 0$. As will be shown later we do this because G loses all of a_l whenever $0^n \notin L_A$. The "shares" a_l and b_i are dynamic and may have their values updated as we bet with them. Therefore, the current capital of G after each bet is $\sum_{i=1}^{\infty}(a_i + b_i)$. Though we describe separately how G bets with a_l and b_i, we may bet with both simultaneously. We bet with some a_l for every stage, but with the b_i's we bet only when the output of the simulated TM is 0. Therefore, every time we bet with b_i we also simultaneously bet with a_l. First let us see how G bets in stage j using the a_l and then with b_i.

Betting with a_l: Our choice of l ensures that $a_l \neq 0$. In fact, a_l will either increase, or reduce to 0 after betting. If we lose a_l in the current stage, then we use $a_{l+1} = 2^{-(l+1)}$ to bet in the next stage. G uses a_l to bet that at least one of the 2^{n_j} tribes that follow 0^{n_j} is completely contained in A; i.e. $0^{n_j} \in L_A$. Call this event $\mathcal{B}_{n_j}$. It is easy to see that for sufficiently large n_j, when strings are included independently in A with probability $1/2$, the probability of event $\mathcal{B}_{n_j}$ is

$$\mathrm{P}_r(\mathcal{B}_{n_j}) = 1 - (1 - 2^{-n_j})^{2^{n_j}} \approx 1 - 1/e.$$

G bets in such a way that whenever the sequence of strings seen satisfies the event $\mathcal{B}_{n_j}$, a_l increases by a factor of approximately $1/(1-1/e)$. If the sequence of strings does not satisfy event $\mathcal{B}_{n_j}$ then G loses all of a_l and will bet with a_{l+1} in the next stage.

We now elaborate on how a_l increases by a factor of approximately $1/(1-1/e)$ when event $\mathcal{B}_{n_j}$ occurs. Let $\omega \in \{0, 1, \star\}^{n_j 2^{n_j}}$ represent the current status of strings in $[0^{n_j}+1, 0^{n_j}+n_j 2^{n_j}]$, $\omega[i]$ indicates the status of string $0^{n_j} + i$, $\star$ indicates the string has not been queried by G

yet, and bits 0 and 1 have their usual meaning. Define

$$G_{a_l}(\omega) = \frac{a_l}{\Pr(\mathcal{B}_{n_j})}\Pr(\mathcal{B}_{n_j}|\omega),$$

where $\Pr(\mathcal{B}_{n_j})$ is the probability a random language satisfies event $\mathcal{B}_{n_j}$, and $\Pr(\mathcal{B}_{n_j}|\omega)$ is the conditional probability of the event $\mathcal{B}_{n_j}$ given the current status of the strings as encoded by ω, i.e. given the strings in $[0^{n_j}+1, 0^{n_j}+n_j2^{n_j}]$ whose membership in A has already been revealed, what is the probability that randomly assigning membership to other strings causes event $\mathcal{B}_{n_j}$ to occur. This probability is rational and easy to compute in $O(2^{2n})$ time by examining the status of the strings in each of the 2^n tribes in $[0^n+1, 0^n+n2^n]$. G_{a_l} is essentially a martingale. Whenever the membership of any string in $[0^{n_j}+1, 0^{n_j}+n_j2^{n_j}]$ is revealed, a_l is then updated to $G_{a_l}(\omega)$. Given $\omega \in \{0,1,\star\}^{n_j2^{n_j}}$ and $b \in \{0,1,\star\}$, let $\omega^{i\to b}$ denote ω with its i^{th} symbol set to b. It is easy to see that

$$G_{a_l}(\omega^{i\to\star}) = \frac{G_{a_l}(\omega^{i\to 0}) + G_{a_l}(\omega^{i\to 1})}{2}.$$

For all sufficiently large n_j,

$$G_{a_l}(\omega) = \frac{a_l}{\mathrm{P}_r(B_{n_j})} \approx a_l/(1-1/e)$$

for any string $\omega \in \{0,1\}^{n_j2^{n_j}}$ that satisfies event $\mathcal{B}_{n_j}$ and 0 for those that do not satisfy $\mathcal{B}_{n_j}$. It is important to note that G can always bet with a_l no matter the order in which it requests the strings in $[0^{n_j}+1, 0^{n_j}+n_j2^{n_j}]$ that it bets on. But as will be shown next the ordering of these strings is important when betting with b_i.

Betting with b_i: Finally, we specify how G bets with "share" b_i which is reserved for betting with M_i. G only bets with b_i when the simulation of M_i on 0^{n_j} returns 0. In this situation G bets that at least $2^{n_j} - (n_j^{\lg i} + i)$ tribes of the 2^{n_j} tribes that follow 0^{n_j} are not contained in A. For simplicity, G does not bet on the tribes that M_i queried. We denote by $\mathcal{C}_{n_j}$ the event that all the tribes not queried by M_i are not contained in A. Event $\mathcal{C}_{n_j}$ occurs with probability at most $(1-2^{-n_j})^{2^{n_j}-(n_j^{\lg i}+i)} \approx 1/e$, and is almost the complement of $\mathcal{B}_{n_j}$. In this case G bets similarly to how it bets with a_l and increases b_i by a factor of

$1/\mathrm{P}_r(\mathcal{C}_{n_j}) \approx e$ whenever the sequence of strings that follow 0^{n_j} satisfies $\mathcal{C}_{n_j}$. If the sequence does not satisfy $\mathcal{C}_{n_j}$ then G loses all of b_i.

We now argue that G succeeds on X. Suppose $A \in X$ and $S \subseteq 0^*$ is the set of input strings G simulates on some TMs in stages $1, 2, \ldots$. Then there are two possibilities:

1. Finitely many strings in S do not belong to L_A,
2. Infinitely many strings in S do not belong to L_A.

Denote by s_k the k^{th} string in S. In the first case, there must be a k such that for every stage $j \geq k$, $s_j \in L_A$. Once we reach stage k, G uses a "share" of its capital $a_i \neq 0$ to bet on s_j belonging to L_A for all $j \geq k$. Therefore, G will increase a_i by a factor of approximately $1/(1 - 1/e)$ for all but finitely many stages $j \geq k$. Therefore, the capital of G will grow without bound in this case.

In the second case, we must reach some stage k at which we use the correct oracle TM M_i that decides L_A on inputs in S. From this stage onward G will never change the TM it simulates on the strings in S we have not seen yet. In this case we are guaranteed this simulation will output 0 infinitely often. It follows by the correctness of M_i and the definition of G that whenever the output of M_i is 0 the "share" of the capital b_i reserved for betting on M_i will be increased by a factor of approximately e. Since this condition is met infinitely often, it follows that the capital of G increases without bound in this case also.

Finally, we show that G can be implemented as a $O(2^{2n})$-betting game; i.e. after $O(2^{2n})$ time, G will have queried all strings of length n. First, we bound the runtime of each round of the betting game; i.e. the time required to bet on a string. This should not be confused with the stages of G which include several rounds of querying. In each round, we have to compute $\sum_{i=1}^{\infty}(a_i + b_i)$ the current capital of G. This sum can easily be computed in $O(2^n)$ time. This is because for each round we change at most two "shares" a_l and b_i to some rational numbers that can be computed in $O(2^n)$ time. Also, the remaining a and b "shares" with indices greater than $l = O(n)$ and $i = O(n)$ respectively retain their initial values, so the sum is easily computable. We may also simulate a TM in each round. Since each

simulated TM M_i has $i \leq n$ it takes $O(n^{\lg n})$ time for the simulation of M_i on 0^n. Therefore, each round is completed in $O(2^n)$ time. After the simulation G requests all the remaining strings in $[0^n + 1, 0^n + n2^n]$ that were not queried during the simulation. Therefore, it takes $O(2^{2n})$ time for G to have requested all strings of length n. □

4.2 Limitations

In this section we examine the possibility of extending Theorem 4.1.1. We show that it cannot be improved to p-random oracles or improved to separate the polynomial-time hierarchy without separating BPP or NP from EXP, respectively. On the other hand, showing that Theorem 4.1.1 cannot be improved to p-random oracles would separate PSPACE from P.

4.2.1 Does $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-random oracle A?

We showed in Theorem 4.1.1 that $\mathrm{P}^A \neq \mathrm{NP}^A$ for a p-betting-game random oracle. It is unknown whether p-betting games and p-martingales are equivalent. If they are, then BPP $\neq$ EXP [11]. This is based on the following theorem and the result that $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete languages for EXP have p-betting-game measure 0 [11].

Theorem 4.2.1 (Buhrman et al. [11])**.** *If the class of $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete languages for* EXP *has p_2-measure zero then* BPP $\neq$ EXP*.*

We show improving Theorem 4.1.1 to p-random oracles would also imply BPP $\neq$ EXP. First, we prove the following for p_2-measure.

Theorem 4.2.2. *If $\{A \mid \mathrm{P}^A \neq \mathrm{NP}^A\}$ has p_2-measure 1, then* BPP $\neq$ EXP*.*

Proof. If L is any $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete language for EXP, then

$$\mathrm{NP}^L \subseteq \mathrm{EXP} \subseteq \mathrm{P}^L \subseteq \mathrm{NP}^L.$$

Therefore, the class of $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete languages for EXP is a subset of $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$. If $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has p_2-measure 0 then so does the class of $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete languages of EXP. Theorem 4.2.1 implies that BPP $\neq$ EXP. □

We have the following for p-random oracles by the universality of p_2-measure for p-measure [31].

Corollary 4.2.3. *If* $P^A \neq NP^A$ *for every* p*-random oracle* A*, then* BPP $\neq$ EXP*.*

Proof. The hypothesis implies that every A with $P^A = NP^A$ is not p-random, i.e. there is a p-martingale that succeeds on A. Let d' be a p_2-martingale that is universal for all p-martingales [31]: $S^\infty[d] \subseteq S^\infty[d']$ for every p-martingale d. Then d' succeeds on $\{A \mid P^A = NP^A\}$. □

4.2.2 Is it possible that $P^A = NP^A$ for some p-random oracle A?

Given Theorem 4.2.2, we consider the possibility of whether $\{A \mid P^A = NP^A\}$ does not have p-measure 0. Because Lutz and Schmidt [36] showed that this class has pspace-measure 0, it turns out that if it does not have p-measure 0, then we have a separation of PSPACE from P.

Theorem 4.2.4 (Lutz and Schmidt [36])**.** *The class* $\{A \mid P^A = NP^A\}$ *has* pspace*-measure 0.*

We note that because every p-betting game may be simulated by a pspace-martingale [11], Theorem 4.2.4 follows as a corollary to Theorem 4.1.1.

Lemma 4.2.5. *If* P = PSPACE*, then for every* pspace*-martingale* d*, there is a* p*-martingale* d' *with* $S^\infty[d] \subseteq S^\infty[d']$*.*

Proof. Let $d : \{0,1\}^* \longrightarrow [0,\infty)$ be a pspace-martingale. Without loss of generality also assume that d is exactly computable [27] and its output is in $\{0,1\}^{\leq p(n)}$, for some polynomial p. Consider the language $L_d = \{\langle w, i, b\rangle \mid \text{the } i\text{th bit of } d(w) \text{ is } b\}$. Clearly $L_d \in$ PSPACE and hence also in P by our hypothesis. We can therefore compute $d(w)$ is polynomial time using L_d. □

Theorem 4.2.6. *If* $\{A \mid P^A = NP^A\}$ *does not have* p*-measure 0, then* P $\neq$ PSPACE*.*

Proof. Assume P = PSPACE. Theorem 4.2.4 and Lemma 4.2.5 imply that $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has p-measure 0. □

Corollary 4.2.7. *If there is a* p*-random oracle* A *such that* $\mathrm{P}^A = \mathrm{NP}^A$*, then* $\mathrm{P} \neq \mathrm{PSPACE}$*.*

4.2.3 Is PH infinite relative to p-betting-game random oracles?

Bennett and Gill [8] showed that $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for a random oracle A, with probability 1. Thus PH^A does not collapse to its first level. Rossman, Servedio, and Tan [46] showed that PH^A is infinite relative to a random oracle, with probability 1.

Can we improve Theorem 4.1.1 to show that PH^A does not collapse for a p-betting-game random oracle? This also has complexity class separation consequences:

Theorem 4.2.8. *For* $k > 0$*, let* $X_k = \{A \mid \sum_k^{\mathrm{P},A} = \prod_k^{\mathrm{P},A}\}$*. If* X_k *has* p_2*-betting-game measure zero, then* $\sum_k^{\mathrm{P}} \neq \mathrm{EXP}$*.*

Proof. We prove the contrapositive. Suppose $\sum_k^{\mathrm{P}} = \mathrm{EXP}$, then $\prod_k^{\mathrm{P}} = \mathrm{EXP}$. Given $A \in \mathrm{EXP}$, then the following containments hold:

$$\Sigma_k^{\mathrm{P}} \subseteq \Sigma_k^{\mathrm{P},A} \subseteq \mathrm{EXP} = \Pi_k^{\mathrm{P}} \subseteq \Pi_k^{\mathrm{P},A} \subseteq \mathrm{EXP} = \Sigma_k^{\mathrm{P}}.$$

Which in turn implies that $\mathrm{EXP} \subseteq X_k$. Since EXP does not have p_2-betting-game measure zero [11] then neither does X_k. Hence, the Theorem follows. □

In particular, we have the following for the first level of PH:

Corollary 4.2.9. *If* $\{A \mid \mathrm{NP}^A \neq \mathrm{coNP}^A\}$ *has p-betting-game measure 1, then* $\mathrm{NP} \neq \mathrm{EXP}$*.*

Because it is open whether betting games have a union lemma [11], it is not clear whether Corollary 4.2.9 may be extended to show that if $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for every p-betting-game random oracle A, then NP ≠ EXP. This extension would hold if there is a p_2-betting game that is universal for all p-betting games. However, we do have the following for p-random oracles.

Corollary 4.2.10. *If* $\mathrm{NP}^A \neq \mathrm{coNP}^A$ *for every* p*-random oracle* A*, then* $\mathrm{NP} \neq \mathrm{EXP}$*.*

Corollary 4.2.11. *If* PH^A *is infinite for every* p*-random oracle* A*, then* $\mathrm{PH} \neq \mathrm{EXP}$*.*

4.3 Conclusion

We have shown that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-betting-game random oracle A (Theorem 4.1.1). Establishing whether this also holds for p-random oracles would imply either BPP $\neq$ EXP (Corollary 4.2.3) or P $\neq$ PSPACE (Corollary 4.2.7). These results, together with Theorems 4.2.4 and 4.2.8, motivate investigating the status of PH relative to pspace-random oracles. In particular:

1. Does $\{A \mid \mathrm{NP}^A = \mathrm{coNP}^A\}$ have pspace-measure 0?

2. More generally, does $\{A \mid \mathrm{PH}^A \text{ collapses}\}$ have pspace-measure 0?

Chapter 5

Complexity Measures of Boolean Functions

In this section we present various results on the nonuniform complexity of languages in uniform complexity classes. The main results of this section are summarized in Table 5.1. Due to the duality between DNFs and CNFs, the DNF in these results can be changed to CNF with very little modifications to the proofs. Similarly, in the case of the influence results we still get the same results after flipping the inequalities ($\leq$ to $\geq$) and changing the range of α from $[0, 1/2)$ to $(1/2, 1]$. The symmetry in the definition of the influence allows us to do so.

Complexity measure	Bound	Measure/Dimension in $R(\Delta)$	Reference
DNF width	$\leq n\left(1-\frac{2\lg\lg n}{\lg n}\right)$, i.o.	measure 0 in P	Theorem 3.1.1
DNF width	$\leq n-\lg(n+\lg^2 n)$, i.o.	measure 0 in E	Theorem 5.1.2
DNF size	$\leq \alpha\frac{2^n}{n}$, i.o. for any $\alpha < 1/2$	measure 0 in E	Theorem 5.2.3
DNF size	$\Theta(\frac{2^n}{\lg n \lg\lg n})$	measure 1 in ESPACE	Theorem 5.2.1
Influence at a point	$\leq \alpha \leq 1/2$, a.e.	dimension $1/2(1+\mathcal{H}(\alpha))$ in E	Theorem 5.3.3
Influence at a point	$\leq \alpha \leq 1/2$, i.o.	dimension $3/4(1+\mathcal{H}(\alpha))$ in E	Theorem 5.3.7
Total influence	$\leq \alpha n$, i.o. for any $\alpha < 1/2$	measure 0 in E	Corollary 5.3.4
DNF-of-parity size	$\Omega(\frac{2^n}{n\lg n})$	measure 1 in EXP	Theorem 5.4.1

Table 5.1: Shows the measure or dimension of languages in $R(\Delta)$ with nonuniform complexity for a given bound.

A common technique used in the s-gales defined in this section is to have them break their initial capital C into infinitely many portions $c_1, c_2, c_3 \cdots$. Each portion will be used to bet only on strings of a fixed length. As long as the portions are easy to compute and their sum converges, the s-gale will never run out of capital to bet with. For clarity in some proofs we omit floor and ceiling functions.

5.1 DNF Width at E

In this this section we present the analogue of Theorem 3.1.1 in E. Moving from P to E allows us to improve the maximum DNF width from $n\left(1 - \frac{2 \lg \lg n}{\lg n}\right)$ to $n - \lg(n + \lg^2 n)$. The theorem follows from a simple application of the following lemma.

Lemma 5.1.1. *Let*

$$X = \{L \in \{0,1\}^\infty | L^{=n} \text{ contains a subcube of dimension } \geq \lg(n + \lg^2 n) \text{ i.o.}\},$$

then X has p*-measure 0 and thus measure 0 in* E*.*

Proof. Consider the martingale d that operates as follows when betting on the prefix strings of some language $L \in \{0,1\}^\infty$.

1. d starts with initial capital 2.
2. d reserves $\frac{1}{n^2}$ of it's initial capital to bet on strings of length n.
3. d further splits the capital $\frac{1}{n^2}$ reserved for betting on length n strings into $2^{n - \lg(n + \lg^2 n)} \binom{n}{\lg(n + \lg^2 n)}$ equal portions.
4. Each portion is reserved for betting that one of the $2^{n - \lg(n + \lg^2 n)} \binom{n}{\lg(n + \lg^2 n)}$ dimension $\lg(n + \lg^2 n)$ subcubes of $\{0,1\}^n$ is contained in $L^{=n}$. I.e. d reserves
$$\frac{2^{\lg(n + \lg^2 n) - n}}{n^2 \binom{n}{\lg(n + \lg^2 n)}}$$
to bet that all of the strings in some particular dimension $\lg(n + \lg^2 n)$ subcube of $\{0,1\}^n$ is contained in $L^{=n}$.

Intuitively, d bets on $x \in \{0,1\}^n$ by using all the capital reserved for betting on dimension-$\lg(n + \lg^2 n)$ subcubes that contain x to bet on $L[x]$ being 1. It doesn't use any of the capital reserved for betting on subcubes that don't contain x.

For $x \in \{0,1\}^n$, $d(L \upharpoonright x)$ can be computed in time polynomial in 2^n. This follows since betting on length n strings only requires we keep track of all $2^{n-\lg(n+\lg^2 n)}\binom{n}{\lg(n+\lg^2 n)} = 2^{O(n)}$ dimension-$\lg(n + \lg^2 n)$ subcubes of $\{0,1\}^n$ and the current capital reserved for each.

Now we show that d succeeds on any $L \in X$. Let $n \in \mathbb{N}$ be such that $L^{=n}$ contains a subcube B of dimension $\lg(n + \lg^2 n)$. Then d must bet on B, therefore, d's capital reserved for B will be doubled when it bets on each of the $2^{\lg(n+\lg^2 n)}$ string contained in B. So it grows from at least

$$\frac{2^{\lg(n+\lg^2 n)-n}}{n^2\binom{n}{\lg(n+\lg^2 n)}}$$

to

$$\frac{2^{\lg(n+\lg^2 n)-n}}{n^2\binom{n}{\lg(n+\lg^2 n)}}2^{2^{\lg(n+\lg^2 n)}} = \frac{2^{\lg^2 n+\lg(n+\lg^2 n)}}{n^2\binom{n}{\lg(n+\lg^2 n)}} = \omega(1)$$

once we finish betting on all the strings in B. Therefore, d's capital grows by a superconstant infinitely often for any $L \in X$, thus d succeeds on X. Therefore, X has p-measure 0 and thus measure 0 in E. □

Theorem 5.1.2. *Let*

$$X = \{L \in \{0,1\}^\infty | L^{=n} \text{ has DNF width } \leq n - \lg(n + \lg^2 n) \text{ i.o.}\}$$

Then X has p*-measure 0 and thus measure 0 in* E*.*

Proof. For any language $L \in \{0,1\}^\infty$, if $L^{=n}$ is computed by a DNF of width $k \geq 0$, then $L^{=n}$ contains a subcube of dimension $n - k$. So the corollary follows by applying Lemma 5.1.1. □

5.2 DNF Size

In this section we extend our previous result about the typical complexity of languages in ESPACE [47]. We showed that almost all languages in ESPACE have DNF complexity close

to $\overline{\ell}(n) = \Theta(\frac{2^n}{\lg n \lg\lg n})$ the expected DNF size of a random Boolean function.

Theorem 5.2.1 (Sekoni [47])**.** *The set of all $x \in \{0,1\}^\infty$ with DNF complexity $D_x(n) = \Theta(\frac{2^n}{\lg n \lg\lg n})$ has measure 1 in* ESPACE*.*

Ideally, we would like to extend Theorem 5.2.1 to a measure result in E. The main technique used in the previous result was compression using space-bounded Kolmogorov complexity. This required us to compute the DNF size of Boolean functions. Due to the inefficiency of known algorithms for computing DNF size the best result we could prove had to be in ESPACE. For our new result we use a different approach with subcubes of Boolean functions rather than their DNF size. We end up with a result in E but with a weaker bound on the DNF size than the previous result.

Lemma 5.2.2. *Let $X = \{L \in \{0,1\}^\infty |$ for some $\alpha < 1/2$, $|L^{=n}| < \alpha 2^n$ i.o.$\}$, then X has* p*-measure 0 and thus measure 0 in* E*.*

We do not present a proof for the previous lemma because its proof is a minor modification of the proof of Lemma 5.1 in [33]. Lemma 5.1 gives a dimension result, but for our purpose we only need a weaker measure result.

Theorem 5.2.3. *Let $X = \{L \in \{0,1\}^\infty | L^{=n}$ has DNF size $\leq \alpha \frac{2^n}{n}$ for some $\alpha < 1/2$ i.o.$\}$ Then X has* p*-measure 0 and thus measure 0 in* E*.*

Proof. Let

$$Y = \{L \in \{0,1\}^\infty | L^{=n} \text{ contains a subcube of dimension } \geq \lg(n + \lg^2 n) \text{ i.o.}\}.$$

We show that $X \cap Y$ and $X \cap Y^{\mathsf{c}}$ both have p-measure 0. Then the theorem follows by the closure of p-measure 0 sets under finite unions.

We now show that $X \cap Y$ has p-measure 0. Lemma 5.1.1 implies that $X \cap Y$, has p-measure 0 since it is the subset of a p-measure 0 set.

To complete the proof we show that $X \cap Y^{\mathsf{c}}$ also has p-measure 0. Let $L \in X \cap Y^{\mathsf{c}}$ and n be such that $L^{=n}$ has DNF size at most $\alpha \frac{2^n}{n}$ and no subcube of dimension up to $\lg(n + \lg^2 n)$.

Then for any rational $\beta \in (\alpha, 1/2)$ and sufficiently large n, $L^{=n}$ has at most

$$\alpha \frac{2^{n+\lg(n+\lg^2 n)}}{n} < \beta 2^n$$

strings since it is covered by at most $\alpha \frac{2^n}{n}$ subcubes and each subcube covers at most $2^{\lg(n+\lg^2 n)}$ strings. By Lemma 5.2.2 the set of all languages with at most $\beta 2^n$ strings of length n for any $\beta < 1/2$ infinitely often has measure 0 in E. Since L was arbitrary it follows that $X \cap Y^{\mathsf{c}}$ has p-measure 0. □

5.3 Influence

The total influence (average sensitivity) is a measure of how sensitive the output of a Boolean function is to a changes in its input. It has been used to study other complexity measures of Boolean functions [7, 43]. In [15] they relate the sensitivity of a Boolean function to its circuit size and formula depth. In this section we study the influence of languages in E. We apply our results to obtain a new bound on the circuit-size complexity of languages in E.

We make a few more definitions for this section.

- dist : $\{0,1\}^n \times \{0,1\}^n \longrightarrow \mathbb{N}$ is the Hamming distance function. dist(x, y) is the number of bit positions at which x and y differ.
- $\mathcal{H} : [0,1] \longrightarrow [0,1]$ is the binary entropy function. It is defined as follows: $\mathcal{H}(\alpha) = 0$ if $\alpha \in \{0,1\}$, and $\mathcal{H}(\alpha) = \alpha \lg \frac{1}{\alpha} + (1-\alpha) \lg \frac{1}{1-\alpha}$ otherwise.
- $\mathcal{H} : (0,1) \times (0,1) \longrightarrow \mathbb{R}$ is a function related to the one given above. It is defined as follows $\mathcal{H}(x, y) = x \lg \frac{1}{y} + (1-x) \lg \frac{1}{1-y}$.

In particular, $\mathcal{H}(x, y)$ is minimized at $y = x$ with value $\mathcal{H}(x)$ and it strictly increases as y moves away from x.

Lemma 5.3.1. *Let* $\text{Inf}[\alpha]_{\text{a.e.}} = \{L \in \{0,1\}^\infty | (\exists i \in [1,n] \text{ s.t. } I_i[L^{=n}] \leq \alpha) \text{ a.e.}\}$. *Then the* p*-dimension of* $\text{Inf}[\alpha]_{\text{a.e.}}$ *is at most* $\frac{1}{2}(1 + \mathcal{H}(\alpha))$ *for* $\alpha \in [0, 1/2]$.

Proof. The proof idea is to design an s-gale which bets that flipping a single bit of a string doesn't change its membership. This s-gale will succeed on any language with sufficiently small influence at a particular coordinate. It succeeds because such a languages will have the same membership for most string pairs of the form $x, x^{\oplus i}$ for some i. Details follow.

Consider the following s-gale $d_{s,y} : \{0,1\}^* \longrightarrow [0,\infty)$ for $s > \frac{1}{2}(1 + \mathcal{H}(\alpha)), y \in [0, 1/2]$, such that 2^s and y are rational numbers. $d_{s,y}$ operates as follows. Just before betting on strings of length n, $d_{s,y}$ divides its current capital into n equal pieces $C_{n,i}$, where $i \in [1, n]$. Each $C_{n,i}$ starts at $d_{s,y}(L \upharpoonright 1^{n-1})/n$. When betting on $x \in \{0,1\}^n$, $d_{s,y}$ wagers $(1-y)C_i$ on the event(s) $f(x) = f(y)$ where $y < x$ and $\{x, y\} \in E_{n,i}$–the set of dimension-i edges in the dimension-n Hamming cube B_n. Intuitively, every bet made by $d_{s,y}$ corresponds to a unique edge in B_n. $C_{n,i}$ is used to bet only on edges in $E_{n,i}$ i.e. $C_{n,i}$ is updated according to bets made on dimension-i edges of B_n. Call a bet bad if it is wrong in its prediction. It is easy to see that the edges of $B_n^{L=n}$ corresponding to bad bets are exactly the dimension-i boundary edges of B_n. We now show that for an appropriately chosen $y, d_{s,y}$ succeeds on any $L \in \mathrm{Inf}[\alpha]_{\text{a.e.}}$. Let n_0 be such that $\forall n \geq n_0\ \exists i$ s.t. $I_i[\chi_{L^{=n}}] \leq \alpha$. Then for any $n \geq n_0$ we must have

$$d_{s,y}(L \upharpoonright 1^n) \geq \frac{d_{s,y}(L \upharpoonright 1^{n-1})}{n}(2^s y)^{\alpha 2^{n-1}}(2^s(1-y))^{(1-\alpha)2^{n-1}}2^{(s-1)2^{n-1}} \tag{5.3.1}$$

$$= \frac{d_{s,y}(L \upharpoonright 1^{n-1})}{n}2^{s2^n}2^{-2^{n-1}}y^{\alpha 2^{n-1}}(1-y)^{(1-\alpha)2^{n-1}} \tag{5.3.2}$$

$$= \frac{d_{s,y}(L \upharpoonright 1^{n-1})}{n}2^{2^n(s-\frac{1}{2}(1+\mathcal{H}(\alpha,y))} \tag{5.3.3}$$

The first inequality holds because when $n \geq n_0$ there is always some $i^\star \in [1, n]$ such that $C_{i^\star}$ starts with capital $\frac{d_{s,y}(L\upharpoonright 1^{n-1})}{n}$ and $E_{n,i^\star}$ has at most $\alpha 2^{n-1}$ boundary edges. Thus our s-gale makes at most $\alpha 2^{n-1}$ bad bets that are accounted for by $(2^s y)^{\alpha 2^{n-1}}$. At least $(1-\alpha)2^{n-1}$ good bets that are accounted for by $(2^s(1-y))^{(1-\alpha)2^{n-1}}$. Finally the 2^{n-1} strings it didn't bet on are accounted for by $2^{(s-1)2^{n-1}}$. Since $s > \frac{1}{2}(1+\mathcal{H}(\alpha))$ and $\lim_{y\to\alpha}\mathcal{H}(\alpha, y) = \mathcal{H}(\alpha)$ we can always choose a rational y sufficiently close to α so that $s > \frac{1}{2}(1 + \mathcal{H}(\alpha, y))$. Therefore, for all $n > n_0$ we can see that the capital of $d_{s,y}$ increases by at least a factor of

$$\frac{2^{2^n(s-\frac{1}{2}(1+\mathcal{H}(\alpha,y))}}{n}$$

after betting on strings in $L^{=n}$. Thus $L \in S^{\infty}[d_{s,y}]$. Our claim follows once we observe that $d_{s,y}$ can easily be implemented in polynomial time. $\square$

Lemma 5.3.2. *Let* $\mathrm{Inf}[\alpha]_{\text{a.e.}} = \{L \in \{0,1\}^{\infty} | (\exists i \in [1,n] \text{ s.t. } I_i[L^{=n}] \leq \alpha) \text{ a.e.}\}$. *Then the* p*-dimension of* $\mathrm{Inf}[\alpha]_{\text{a.e.}}$ *is at least* $\frac{1}{2}(1 + \mathcal{H}(\alpha))$ *for* $\alpha \in [0, 1/2]$.

Proof. The proof idea is to construct a language L in both $\mathrm{Inf}[\alpha]_{\text{a.e.}}$ and $R(\mathrm{p}) = \mathrm{E}$ such that, any given p-computable s-gale d_s will fail to succeed on L whenever $s < \frac{1}{2}(1 + \mathcal{H}(\alpha))$. Let $L_0^{=n}$ and $L_1^{=n}$ denote the first 2^{n-1} and the last 2^{n-1} bits of $L^{=n} \in \{0,1\}^{2^n}$ respectively. Note that $L_0^{=n}$ ($L_1^{=n}$) is composed of the the decision bits for the strings of length n that begin with bit 0 (1). We construct L to be a language with low influence at the first bit position almost everywhere. We do this by choosing the bits of $L_0^{=n}$ to minimize d_s. Then the bits of $L_1^{=n}$ will be chosen to also minimize d_s but with the additional constraint that the function encoded by concatenating $L_0^{=n}$ and $L_1^{=n}$ (i.e. $L^{=n}$) has influence at the first coordinate at most α. This is done by ensuring that the hamming distance between $L_0^{=n}$ and $L_1^{=n}$ is at most $\alpha 2^n$. Details follow.

We show that $\dim_p(\mathrm{Inf}[\alpha]_{\text{a.e.}}) \geq \frac{1}{2}(1+\mathcal{H}(\alpha))$ by constructing a language $L \in (\mathrm{Inf}[\alpha]_{\text{a.e.}} \cap \mathrm{E}) \setminus S^{\infty}[d_s]$ for any s-*gale* with $s < \frac{1}{2}(1 + \mathcal{H}(\alpha))$. For natural numbers $c_1 < c_2$ we define a language L using the constructor $\delta_{c_1,c_2} : \{0,1\}^* \longrightarrow \{0,1\}^*$. First we specify how it constructs the bits of $L_0^{=n}$, then the bits of $L_1^{=n}$.

Constructing $L_0^{=n}$:

$$\delta_{c_1,c_2}(w) = \begin{cases} 1, & w = \lambda \\ w1, & \text{the first bit of } s_{|w|+1} \text{ is 0 and } d_s(w1) \leq d_s(w0) \\ w0, & \text{the first bit of } s_{|w|+1} \text{ is 0 and } d_s(w0) < d_s(w1) \end{cases}.$$

Before we specify $L_1^{=n}$ i.e. the cases where the first bit of $s_{|w|+1}$ is 1, we make a few definitions. For $m, k \in \mathbb{N}, w \in \{0,1\}^m$ let

$$B_{m,k,w} = \{u \in \{0,1\}^m | \mathrm{dist}(u,w) = k\}.$$

I.e. $B_{m,k,w}$ is the set of strings of length m that differ from w in k coordinates.

Now we specify how δ_{c_1,c_2} constructs the bits of $L_1^{=n}$. In this case the first bit of $s_{|w|+1}$ is 1. Let $n = |s_{|w|+1}|$, $m = c_2 n$ and $k = c_1 n$ and $x = w[s_{|w|+1}^{1\mapsto 0}, s_{|w|+m}^{1\mapsto 0}]$. If $s_{|w|+m}$ has length n, then in the standard lexicographic order δ_{c_1,c_2} finds the first string u in $B_{m,k,x}$ that minimizes $d(wu)$ and then outputs wu. Otherwise, $s_{|w|+m}$ does not have length n. In this case δ_{c_1,c_2} outputs $w[s_{|w|+1}^{1\mapsto 0}, 01^{n-1}]$. Since $n = O(\lg |w|)$ it is easy to see that δ_{c_1,c_2} is is polynomial-time computable. Also, by the definition of δ_{c_1,c_2}, we see the influence of $L^{=n}$ at coordinate 1 is $(\lfloor 2^n/c_2 \rfloor)c_1/2^n$. Thus, the influence of L at coordinate 1 is at most c_1/c_2 almost everywhere. Next we show that there exist $c_1, c_2 \in \mathbb{N}$ for which d_s doesn't succeed on the constructed language $R(\delta_{c_1,c_2})$. First we see by the action of δ_{c_1,c_2} on strings whose first bit is 0 that $d_s(L \upharpoonright 01^{n-1}) \leq d_s(L \upharpoonright 1^{n-1})2^{(s-1)2^{n-1}}$ for every n.

Similarly, we see that $d_s(L \upharpoonright 1^n) \leq d_s(L \upharpoonright 01^{n-1})2^{(s-\mathcal{H}(c_1/c_2))\lfloor 2^{n-1}/m \rfloor m + sr}$ where $0 \leq r < m$. This follows using the approximation $e(n/e)^n \leq n! \leq en(n/e)^n$ to show that for sufficiently large n and any $w \in \{0,1\}^m$,

$$|B_{m,k,w}| = \binom{m}{k} > 2^{m\mathcal{H}(m/k)} = 2^{m\mathcal{H}(c_1/c_2)}.$$

By a simple averaging argument (Corollary 3.4 in [33]) we see that for any subset of $\{0,1\}^m$ with more than $2^{\mathcal{H}(c_1/c_2)m}$ elements there is at least one $u \in \{0,1\}^m$ with $d(wu) \leq 2^{(s-\mathcal{H}(c_1/c_2))}d(w)$ for any $w \in \{0,1\}^*$. Thus by combining the two inequalities we get

$$d_s(L \upharpoonright 1^n) \leq d_s(L \upharpoonright 1^{n-1})2^{(s-1)2^{n-1}}2^{(s-\mathcal{H}(c_1/c_2))\lfloor 2^{n-1}/m \rfloor m + sr} \tag{5.3.4}$$

$$= d_s(L \upharpoonright 1^{n-1})2^{2^n(s-\mathcal{H}(c_1/c_2)/2-1/2-o(1))} \tag{5.3.5}$$

Thus for any $s < \frac{1}{2}(1 + \mathcal{H}(\alpha))$ we can find $c_1, c_2 \in \mathbb{N}$ with c_1/c_2 sufficiently close to α so that $s < \frac{1}{2}(1 + \mathcal{H}(c_1/c_2))$. Then by inequality 5.3.5 for sufficiently large n any d_s will cease to increase i.e. $L \notin S^\infty[d_s]$. Therefore, $L \in (\mathrm{Inf}[\alpha]_{\text{a.e.}} \cap \mathrm{E}) \setminus S^\infty[d_s]$, this implies that $\dim_p(\mathrm{Inf}[\alpha]_{\text{a.e.}}) \geq \frac{1}{2}(1 + \mathcal{H}(\alpha))$ since s is arbitrary. □

Theorem 5.3.3. *Let* $\mathrm{Inf}[\alpha]_{\text{a.e.}} = \{L \in \{0,1\}^\infty | (\exists i \in [1,n] \text{ s.t. } I_i[L^{=n}] \leq \alpha) \text{ a.e.}\}$. *Then the* p*-dimension of* $\mathrm{Inf}[\alpha]_{\text{a.e.}}$ *is* $\frac{1}{2}(1 + \mathcal{H}(\alpha))$.

Proof. Follows from Lemma 5.3.1 and 5.3.2. □

Corollary 5.3.4. *Let* $\text{TotalInf}[\alpha]_{\text{a.e.}} = \{L \in \{0,1\}^\infty | I[L^{=n}] \leq \alpha n \text{ a.e.}\}$. *Then the* p*-measure of* $\text{TotalInf}[\alpha]_{\text{a.e.}}$ *is 0 for* $\alpha \in [0, 1/2]$.

Proof. Since $I[L^{=n}] = \sum_{i=1}^{n} I_i[L^{=n}]$ for any language L, it follows that $\exists i \in [1,n]$ s.t. $I_i[L^{=n}] \leq \alpha$ whenever $I[L^{=n}] \leq \alpha n$. Therefore, any $L \in \text{TotalInf}[\alpha]_{\text{a.e.}}$ also belong to $\text{Inf}[\alpha]_{\text{a.e.}}$. By Theorem 5.3.3 the dimension of $\text{Inf}[\alpha]_{\text{a.e.}}$ must be less than 1 and thus must have p-measure 0. Therefore, $\text{TotalInf}[\alpha]_{\text{a.e.}}$ also has p-measure 0 and the result follows. □

Lemma 5.3.5. *Let* $\text{Inf}[\alpha]_{\text{i.o.}} = \{L \in \{0,1\}^\infty | (\exists i \in [1,n] \text{ s.t. } I_i[L^{=n}] \leq \alpha) \text{ i.o.}\}$. *Then the* p*-dimension of* $\text{Inf}[\alpha]_{\text{i.o.}}$ *is at most* $\frac{3}{4}(1 + \mathcal{H}(\alpha))$ *for* $\alpha \in [0, 1/2]$.

Proof. The proof is similar to that of Lemma 5.3.1 so we just note the main differences here. For $s > \frac{3}{4}(1 + \mathcal{H}(\alpha))$ and $y \in [0, 1/2]$ such that 2^s and y are rational numbers, we define $d_{s,y} : \{0,1\}^* \longrightarrow [0, \infty)$ just as in Lemma 5.3.1 with the following exception. $d_{s,y}$ no longer reserves $C_{n,i} = d_{s,y}(L \upharpoonright 1^{n-1})$ for betting on dimension-i edges. Instead it reserves $1/n^3$ of its initial capital for betting on dimension-i edges. Therefore, when $d_{s,y}$ bets on strings of length n it has $C_{n,i} = 2^{(2^n-1)(s-1)}/n^3$ available for betting on dimension-i edges of B_n. Let n be such that there is an $i \in [1,n]$ with $I_i[L^{=n}] \leq \alpha$, then

$$d_{s,y}(L \upharpoonright 1^n) \geq \frac{2^{(2^n-1)(s-1)}}{n^3} (2^s y)^{\alpha 2^{n-1}} (2^s(1-y))^{(1-\alpha)2^{n-1}} 2^{(s-1)2^{n-1}} \tag{5.3.6}$$

$$= \frac{2^{(2^n-1)(s-1)}}{n^3} 2^{s2^{n-1}} y^{\alpha 2^{n-1}} (1-y)^{(1-\alpha)2^{n-1}} 2^{(s-1)2^{n-1}} \tag{5.3.7}$$

$$= 2^{2^{n-1}(4s-3-\mathcal{H}(\alpha,y)-o(1))} \tag{5.3.8}$$

Since $s > \frac{3}{4}(1 + \mathcal{H}(\alpha))$ and $\lim_{y \to \alpha} \mathcal{H}(\alpha, y) = \mathcal{H}(\alpha)$ we can always choose a rational y sufficiently close to α such that $s > \frac{3}{4}(1 + \mathcal{H}(\alpha, y))$. Then $d_{s,y}(L \upharpoonright 1^n)$ is unbounded for infinitely many n, thus $L \in S^\infty[d_{s,y}]$. Since s was arbitrary it follows that $\dim_p(\text{Inf}[\alpha]_{\text{i.o.}}) \leq \frac{3}{4}(1 + \mathcal{H}(\alpha))$. □

Lemma 5.3.6. *Let* $\text{Inf}[\alpha]_{\text{i.o.}} = \{L \in \{0,1\}^\infty | (\exists i \in [1,n] \text{ s.t. } I_i[L^{=n}] \leq \alpha) \text{ i.o.}\}$. *Then the* p*-dimension of* $\text{Inf}[\alpha]_{\text{i.o.}}$ *is at least* $\frac{1}{4}(3 + \mathcal{H}(\alpha))$ *for* $\alpha \in [0, 1/2]$.

Proof. We now construct a language $L \in (\text{Inf}[\alpha]_{\text{i.o.}} \cap \text{E}) \setminus S^\infty[d_s]$ for any p-computable s-gale, where $s < \frac{3}{4}(1 + \mathcal{H}(\alpha))$ and $\alpha \in [0, 1/2]$. We define a polynomial time constructor

$\delta^{\text{i.o.}}_{c_1,c_2} : \{0,1\}^* \longrightarrow [0,\infty)$. $\delta^{\text{i.o.}}_{c_1,c_2}$ behaves like the constructor δ_{c_1,c_2} in Lemma 5.3.2 on string lengths that are perfect squares, otherwise it minimizes d_s i.e.

$$\delta^{\text{i.o.}}_{c_1,c_2}(w) = \begin{cases} \delta_{c_1,c_2}(w), & |s_{|w|+1}| = n^2, n \in \mathbb{N} \\ w1, & d_s(w1) \leq d_s(w0) \\ w0, & d_s(w0) < d_s(w1) \end{cases}$$

Clearly $\delta^{\text{i.o.}}_{c_1,c_2}$ is polynomial time computable. Also note that for the constructed language L the influence at coordinate 1 of $L^{=n}$ is at most c_1/c_2 whenever n is a perfect square.

Now we show that d_s doesn't succeed on L. By the definition of δ_{c_1,c_2}, d_s doesn't increase when betting on strings whose lengths aren't perfect squares. Therefore, we only need to consider strings of length n^2. Let $x \in \{0,1\}^{n^2}$ then

$$d_s(L \upharpoonright 1^{n^2}) = d_{s,y}(L \upharpoonright 0^{n^2-1}) 2^{2^{n^2}(s-\mathcal{H}(c_1/c_2)/2-1/2-o(1))} \tag{5.3.9}$$

$$= 2^{2^{n^2}(s-1-o(1))} 2^{2^{n^2}(s-\mathcal{H}(c_1/c_2)/2-1/2-o(1))} \tag{5.3.10}$$

$$= 2^{2^{n^2}(2s-\mathcal{H}(c_1/c_2)/2-3/2-o(1))} \tag{5.3.11}$$

$d_{s,y}(L \upharpoonright 0^{n^2-1}) = 2^{2^{n^2}(s-1-o(1))}$ follows because our definition $\delta^{\text{i.o.}}_{c_1,c_2}$ doesn't allow d_s to grow when betting on strings whose lengths aren't a perfect square.

Therefore, if we choose c_1, c_2 with c_1/c_2 sufficiently close α so that $s < \frac{1}{4}(3 + \mathcal{H}(c_1/c_2))$, then d_s will not succeed on L. Since s was arbitrary it follows that $\dim_p(\text{Inf}[\alpha]_{\text{i.o.}}) \geq \frac{3}{4}(1 + \mathcal{H}(\alpha))$. □

Theorem 5.3.7. *Let* $\text{Inf}[\alpha]_{\text{i.o.}} = \{L \in \{0,1\}^\infty | (\exists i \in [1,n] \text{ s.t. } I_i[L^{=n}] \leq \alpha) \text{ i.o.}\}$. *Then the* p*-dimension of* $\text{Inf}[\alpha]_{\text{i.o.}}$ *is* $\frac{3}{4}(1 + \mathcal{H}(\alpha))$ *for* $\alpha \in [0, 1/2]$.

Proof. Follows from Lemma 5.3.5 and 5.3.6. □

Corollary 5.3.8. *Let* $\text{TotalInf}[\alpha]_{\text{i.o.}} = \{L \in \{0,1\}^\infty | I[L^{=n}] \leq \alpha n \text{ i.o.}\}$. *Then the* p*-measure of* $\text{TotalInf}[\alpha]_{\text{i.o.}}$ *is 0 for* $\alpha \in [0, 1/2]$ *such that* $\mathcal{H}(\alpha) < 1/3$.

Proof. Since $I[L^{=n}] = \sum_{i=1}^n I_i[L^{=n}]$ for any language, it follows that $\exists i \in [1,n]$ s.t. $I_i[L^{=n}] \leq \alpha$ whenever $I[L^{=n}] \leq \alpha n$. Therefore, any $L \in \text{TotalInf}[\alpha]_{\text{i.o.}}$ also belong to $\text{Inf}[\alpha]_{\text{i.o.}}$. By

Theorem 5.3.7 and our hypothesis that $\mathcal{H}(\alpha) < 1/3$ the dimension of $\text{Inf}[\alpha]_{\text{i.o.}}$ must be less than 1 and thus must have p-measure 0. Therefore, $\text{TotalInf}[\alpha]_{\text{i.o.}}$ also has p-measure 0. □

We present a simple application of the previous results to determine the measure of a family of shallow subexponential sized circuits in E. The following theorem due to Cai et al [12], shows that almost all languages in P don't have shallow subexponential size circuits. A simple application of the following two theorems along with Theorem 5.3.7 yields an analogous result in E.

Theorem 5.3.9 (Cai et al. [12])**.** *For all $\delta > 0$ and all sufficiently large n, the collection of all languages recognized by nonuniform families of circuits with AND, OR, and NOT gates that have depth $d(n) = o(\lg n / \lg \lg n)$ and size $2^{n^{\frac{1}{2d(n)+6+\delta}}}$ infinitely often has measure 0 in* P*.*

Theorem 5.3.10 (Boppana [10])**.** *Circuits of depth d and size s have total influence (average sensitivity) $O(\lg s)^{d-1}$.*

Theorem 5.3.11. *For all $\delta > 0$ and all sufficiently large n, the collection of all languages recognized by nonuniform families of circuits with AND, OR, and NOT gates that have depth $d(n) = o(\lg n)$ and size $2^{n^{\frac{1}{d(n)(1+\delta)}}}$ infinitely often has measure 0 in* E*.*

Proof. By Theorem 5.3.10, the total influence of the circuit in the statement of this theorem is $o(n^{\frac{1}{1+\delta}})$. Therefore, the set of all languages described in the statement of the theorem must also have very low influence ($o(1)$) at some coordinate infinitely often. This follows because the total influence is the sum of the influence at each of the n coordinates. By Theorem 5.3.7 the set of such languages has dimension < 1 in E, and thus measure 0 in E. □

5.4 DNF of Parities

DNF-of-parity circuits are a generalization of DNF formulas. They have been studied by Jukna [28, 29], Grolmusz [16], and Cohen et al. [13]. They proved lower bounds for the complexity of various Boolean functions in this model. In particular, Chohen et al. showed

that random Boolean functions have DNF-of-parity complexity in $\Omega(\frac{2^n}{n \lg n})$. Our result qualitatively strengthens theirs to p_2-randomness. We show that the DNF-of-parity complexity of almost ever language in EXP is $\Omega(\frac{2^n}{n \lg n})$.

For the main result (Theorem 5.4.1) we make a few more definitions. We will view $\{0,1\}^n$ as the vector space $\mathbb{F}^n$ over the two element field $\mathbb{F}$.

An affine subspace is a translation of a vector space by a vector. If V is a subspace of $\{0,1\}^n$, then $x + V$ is an affine subspace. It is easy to see that there are at most $2^n \binom{2^n}{k}$ dimension k affine subspaces of $\{0,1\}^n$. For our purposes this trivial approximation is sufficient. We encode a dimension k affine subspace $x + V$ by the k-tuple $(x, v_1, \cdots, v_k)$, where the v_is are a basis of V.

A Boolean function $f : \{0,1\}^n \longrightarrow \{0,1\}$ is an affine disperser for dimension $k \leq n$, if f is not constant whenever it is restricted to any dimension k affine subspace of $\{0,1\}^n$. We denote by $\mathrm{DNF}_\oplus(f)$ the DNF-of-parity size of a Boolean function f [13].

Theorem 5.4.1. *The set of all languages with* $\mathrm{DNF}_\oplus$ *size* $\Omega(2^n/n \lg n)$ *has* p_2*-measure 1 and thus measure 1 in* EXP.

The proof of Theorem 5.4.1 is lengthy but the idea is simple. The key observation is that most functions with low $\mathrm{DNF}_\oplus$ size must be constant on a large enough affine subspace. We define a martingale by Algorithm 5.4 that bets and succeeds in this situation.

Algorithm 5.4.

Input: $1 \leq k \leq n, w = L \restriction x$, where $x \in \{0,1\}^n$
Output: $d_k(w) \in [0, \infty)$
if $n = 0$ **then**
  return 4
end if
$N = 2^n \binom{2^n}{k}$
$C_0 = (\frac{1}{n^2 N}, \frac{1}{n^2 N}, \cdots, \frac{1}{n^2 N}) \in (\mathbb{R}^+ \cup \{0\})^N$
$C_1 = (\frac{1}{n^2 N}, \frac{1}{n^2 N}, \cdots, \frac{1}{n^2 N}) \in (\mathbb{R}^+ \cup \{0\})^N$
for String $s = 0^n$ to x **do**
  for String $v = 0^n$ to 1^n **do**
    for each k-set $B \in \binom{\{0,1\}^n}{k}$ **do**
      if $v + B$ is an affine subspace and contains s **then**
        $C_0[v][B] = 2(1 - L[s])C_0[v][B]$
        $C_1[v][B] = 2L[s]C_1[v][B]$
      end if
    end for
  end for
end for
return $\sum_{[v][B]}(C_0[v][B] + C_1[v][B])$

Note that the arrays C_0 and C_1 created by the algorithm are indexed by a length n string and a set of k length n strings. The indices of these arrays encode an affine subspace whenever the the set of k strings is a basis. $\binom{\{0,1\}^n}{k}$ denotes the subset of the power set of $\{0,1\}^n$ that contains only sets with k elements.

The main idea behind Algorithm 5.4 is to implement a martingale that bets on its input not being an affine disperser for dimension k. i.e. the martingale bets that the Boolean function $L^{=n}$ is constant on at least one of its dimension k affine subspaces. The martingale splits its initial capital 4 into infinitely many portions. When betting on strings of length n the martingale only risks $2/n^2$ of its initial capital. Since there are at most $2^n \binom{2^n}{k}$ dimension k affine subspaces of $\{0,1\}^n$, and $L^{=n}$ can have at most two values on each subspace, the martingale splits its capital into $2^{n+1}\binom{2^n}{k}$ portions. Each portion is reserved for betting that the value of $L^{=n}$ is constant with value 0 or 1 on some dimension k affine subspace.

We now argue with the following claims that Algorithm 5.4 does implement a p_2-martingale.

Claim. *For any non-negative integer $k \leq n$ and $w \leq 2^{n+1} - 1$, Algorithm 5.4 runs in time $O(2^{n^2})$.*

Proof. We first note that all the numbers used by the algorithm are rational and can easily encoded by $O(n^2)$ bits. Therefore, adding and multiplying these numbers can be implement in time polynomial in n. We can therefore ignore these operations since we only need the runtime of the algorithm to be quasi-polynomial in 2^n i.e. the algorithm is in p_2.

The assignments in lines 6 to 8 can be implemented in $O(2^{n^2})$. The for-loops of lines 9 to 11 are straightforward to implement. The condition of the If statement in line 12 can be evaluated in $O(n)$ with elementary linear algebra. Therefore, the nested for-loops take $O(2^{n^2})$. Adding the time for line 19 doesn't change the class of the run time, so the claim follows. □

As we will show later the martingale implemented by the algorithm succeeds on all languages in which $L^{=n}$ is not an affine disperser for some dimension $k \in O(\lg n + \lg\lg n + O(1))$ infinitely often.

Claim. *Algorithm 5.4 is a martingale.*

Proof. C_0 and C_1 can be viewed as $2^{n+1}\binom{2^n}{k}$ martingales running in parallel. Lines 13 and 14 ensure that the averaging property of each martingale is preserved. The variable C is the sum of the values of the martingales. Since the sum of martingales is also a martingale, Algorithm 5.4 is also a martingale. □

Lemma 5.4.2. *Let $d_m()$ denote the output of Algorithm 5.4 with input k set to m. Consider the function $g(n)$ defined as follows:*

$$g(n) = \begin{cases} 1, & \lg n + \lg\lg n \in \mathbb{N} \\ \lceil \lg n + \lg\lg n \rceil - \lg n + \lg\lg n, & \text{otherwise} \end{cases}$$

Let $\mathrm{AD} = \left\{ L \in \{0,1\}^\infty \middle| L^{=n} \text{ is not an affine disperser for dimension } \lg n + \lg\lg n + g(n) \text{ i.o.} \right\}$ *Then $d_{\lg n + \lg\lg n + g(n)}()$ succeeds on* AD*. In particular AD has* p_2*-measure 0.*

Proof. Note that $\lg n + \lg\lg n + g(n)$ is simply the smallest integer greater than $\lg n + \lg\lg n$. Let $L \in \text{AD}$ and $n \in \mathbb{N}$ such that $L^{=n}$ isn't an affine disperser for dimension $\lg n + \lg\lg n + g(n)$. Then there must exist some dimension $\lg n + \lg\lg n + g(n)$ affine subspace V of $\{0,1\}^n$ that $L^{=n}$ is constant on. Our martingale $d_k()$, with $k = \lg n + \lg\lg n + g(n)$, defined in Algorithm 5.4 uses the arrays C_0 and C_1 to reserve $\frac{1}{n^2 2^{n+1}\binom{2^n}{k}}$ of its initial capital to bet on $L^{=n}$ having value $b \in \{0,1\}$ on each dimension $\lg n + \lg\lg n + g(n)$ affine subspace of $\{0,1\}^n$. Since $L^{=n}$ is constant on V, the capital reserved for V grows from

$$\frac{1}{n^2 2^{n+1}\binom{2^n}{k}}$$

to

$$\frac{2^{2^k}}{n^2 2^{n+1}\binom{2^n}{k}} = \frac{2^{2^{g(n)} n \lg n}}{n^2 2^{n+1}\binom{2^n}{\lg n + \lg\lg n + g(n)}} = \omega(1)$$

after betting on all strings in V. Therefore, $d_k()$ with $k = \lg n + \lg\lg n + g(n)$ succeeds on AD since there are infinitely many n such that $L^{=n}$ isn't an affine disperser for dimension $\lg n + \lg\lg n + g(n)$. □

Lemma 5.4.3 (Cohen et al. [13])**.** *Let $f : \{0,1\}^n \longrightarrow \{0,1\}$ be an affine disperser for dimension k. Then $DNF_{\oplus}(f) \geq |f^{-1}(1)|2^{1-k}$.*

We are now ready to prove Theorem 5.4.1, it follows directly from the next lemma.

Lemma 5.4.4. *For $0 \leq \delta < 1/2$, let*

$$X = \left\{ L \in \{0,1\}^\infty \middle| DNF_{\oplus}(L^{=n}) \leq \delta \frac{2^n}{n \lg n} \text{ i.o.} \right\}$$

Then X has p_2-measure 0 and thus measure 0 in EXP*.*

Proof. By Lemma 5.2.2, Lemma 5.4.2, and the closure of measure 0 sets under finite unions, we know that the set of languages $L \in X$ that satisfies any of the following two properties has p_2 measure 0.

1. For any $\alpha < 1/2$, the number of strings in $L^{=n}$ is at most $\alpha 2^n$ infinitely often.

2. $L^{=n}$ is not an affine disperser for dimension $\lg n + \lg\lg n + g(n)$ infinitely often.

Therefore, we only need to consider languages $L \in X$ that don't satisfy the previously mentioned properties. Suppose L is such a language, then for any $\alpha < 1/2$ and sufficiently large n, $|L^{=n}| > \alpha 2^n$ and will be an affine disperser for dimension $\lg n + \lg\lg n + g(n)$ for all but finitely many n. This implies that

$$DNF_{\oplus}(L^{=n}) \geq \frac{\alpha 2^n}{2^{\lg n + \lg\lg n + g(n) - 1}}$$

for all but finitely many n. This follows from Lemma 5.4.3. Since $0 < g(n) \leq 1$, it follows that

$$DNF_{\oplus}(L^{=n}) \geq \frac{\alpha 2^n}{n \lg n}$$

for any $\alpha < 1/2$ and all but finitely many n. Therefore, L cannot belong to X and the Lemma follows. □

5.5 Conclusion

The obvious direction for extending the results presented in this section is to consider other complexity measures of Boolean functions. For example, studying the noise sensitivity [44] of languages would be a natural extension of our results on the influence of languages. Another direction is to extend our results to stronger notions of measure. For example, the bottleneck of the p_2-martingale that succeeds on languages with low DNF-of-parity complexity is a counting problem. If we could find a more efficient way of counting or approximating the number of affine subspaces that are consistent with the revealed bits of a language, then our result could be improved from measure in EXP to measure in E.

References

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

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

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

[4] E. Allender and M. Strauss. Measure on P: Robustness of the notion. In *International Symposium on Mathematical Foundations of Computer Science*, pages 129–138. Springer, 1995.

[5] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley & Sons, 2016.

[6] K. Ambos-Spies and E. Mayordomo. Resource-bounded measure and randomness. In A. Sorbi, editor, *Complexity, Logic and Recursion Theory*, Lecture Notes in Pure and Applied Mathematics, pages 1–47. Marcel Dekker, New York, N.Y., 1997.

[7] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[8] C. H. Bennett and J. Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1. *SIAM Journal on Computing*, 10:96–113, 1981.

[9] R. V. Book, J. H. Lutz, and K. W. Wagner. An observation on probability versus randomness with applications to complexity classes. *Mathematical Systems Theory*, 27:201–209, 1994.

[10] R. B. Boppana. The average sensitivity of bounded-depth circuits. *Information processing letters*, 63(5):257–261, 1997.

[11] H. Buhrman, D. van Melkebeek, K. W. Regan, D. Sivakumar, and M. Strauss. A generalization of resource-bounded measure, with application to the BPP vs. EXP problem. *SIAM Journal on Computing*, 30(2):576–601, 2001.

[12] J. Cai, D. Sivakumar, and M. Strauss. Constant-depth circuits and the Lutz hypothesis. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, pages 595–604. IEEE Computer Society, 1997.

[13] G. Cohen and I. Shinkar. The complexity of DNF of parities. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 47–58. ACM, 2016.

[14] Y. Crama and P. L. Hammer. *Boolean Functions - Theory, Algorithms, and Applications*, volume 142 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 2011.

[15] P. Gopalan, N. Nisan, R. A. Servedio, K. Talwar, and A. Wigderson. Smooth boolean functions are easy: Efficient algorithms for low-sensitivity functions. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 59–70. ACM, 2016.

[16] V. Grolmusz. A weight-size trade-off for circuits with mod m gates. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 68–74. ACM, 1994.

[17] R. C. Harkins and J. M. Hitchcock. Exact learning algorithms, betting games, and circuit lower bounds. *ACM Transactions on Computation Theory*, 5(4):article 18, 2013.

[18] J. Håstad. *Computational Limitations for Small-Depth Circuits*. The MIT Press, 1986.

[19] J. M. Hitchcock. Hausdorff dimension and oracle constructions. *Theoretical Computer Science*, 355(3):382–388, 2006.

[20] J. M. Hitchcock. Online learning and resource-bounded dimension: Winnow yields new lower bounds for hard sets. *SIAM Journal on Computing*, 36(6):1696–1708, 2007.

[21] J. M. Hitchcock and A. Pavan. Comparing reductions to NP-complete sets. *Information and Computation*, 205(5):694–706, 2007.

[22] J. M. Hitchcock and A. Sekoni. Nondeterminisic sublinear time has measure 0 in P. *CoRR*, abs/1801.05884, 2018.

[23] J. M. Hitchcock, A. Sekoni, and H. Shafei. Polynomial-time random oracles and separating complexity classes. *CoRR*, abs/1801.07317, 2018.

[24] J. M. Hitchcock and Hadi Shafei. Autoreducibility of NP-complete Sets under Strong Hypotheses. *Computational Complexity*, 27(1):63–97, 2018.

[25] J. M. Hitchcock and N. V. Vinodchandran. Dimension, entropy rates, and compression. *Journal of Computer and System Sciences*, 72(4):760–782, 2006.

[26] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220–229. ACM, 1997.

[27] D. W. Juedes and J. H. Lutz. Weak completeness in E and E_2. *Theoretical Computer Science*, 143(1):149–158, 1995.

[28] S. Jukna. On graph complexity. *Combinatorics, Probability and Computing*, 15(6):855–876, 2006.

[29] S. Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

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

[31] J. H. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences*, 44(2):220–258, 1992.

[32] J. H. Lutz. The quantitative structure of exponential time. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 225–254. Springer-Verlag, 1997.

[33] J. H. Lutz. Dimension in complexity classes. *SIAM Journal on Computing*, 32(5):1236–1259, 2003.

[34] J. H. Lutz and E. Mayordomo. Cook versus Karp-Levin: Separating completeness notions if NP is not small. *Theoretical Computer Science*, 164(1–2):141–163, 1996.

[35] J. H. Lutz and E. Mayordomo. Twelve problems in resource-bounded measure. *Bulletin of the European Association for Theoretical Computer Science*, 68:64–80, 1999. Also in *Current Trends in Theoretical Computer Science: Entering the 21st Century*, pages 83–101, World Scientific Publishing, 2001.

[36] J. H. Lutz and W. J. Schmidt. Circuit size relative to pseudorandom oracles. *Theoretical Computer Science*, 107(1):95–120, March 1993.

[37] P. Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[38] W. Merkle, J. S. Miller, A. Nies, J. Reimann, and F. Stephan. Kolmogorov-Loveland randomness and stochasticity. *Annals of Pure and Applied Logic*, 138(1–3):183–210, 2006.

[39] P. Moser. Martingale families and dimension in P. *Theoretical Computer Science*, 400(1-3):46–61, 2008.

[40] A. A. Muchnik, A. L. Semenov, and V. A. Uspensky. Mathematical metaphysics of randomness. *Theoretical Computer Science*, 207(2):263 – 317, 1998.

[41] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of computer and System Sciences*, 49(2):149–167, 1994.

[42] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[43] R. O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69:68–94, 2004.

[44] R. O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014.

[45] N. Pippenger. The shortest disjunctive normal form of a random Boolean function. *Random Structures & Algorithms*, pages 161–186, 2003.

[46] B. Rossman, R. A. Servedio, and L. Tan. An average-case depth hierarchy theorem for boolean circuits. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1030–1048. IEEE, 2015.

[47] A. Sekoni. Polynomial-space randomness and DNF complexity. Master's thesis, University of Wyoming, 2014.

[48] C. Shannon et al. The synthesis of two-terminal switching circuits. *Bell Labs Technical Journal*, 28(1):59–98, 1949.

[49] M. Strauss. Measure on P: Strength of the notion. *Information and Computation*, 136(1):1–23, 1997.