Polynomial-Time Random Oracles and Separating Complexity Classes

John M. Hitchcock * Adewale Sekoni † Hadi Shafei ‡

Abstract

Bennett and Gill (1981) showed that $P^A \neq NP^A \neq coNP^A$ for a random oracle A, with probability 1. We investigate whether this result extends to individual polynomial-time random oracles. We consider two notions of random oracles: p-random oracles in the sense of martingales and resource-bounded measure (Lutz, 1992; Ambos-Spies et al., 1997), and p-betting-game random oracles using the betting games generalization of resource-bounded measure (Buhrman et al., 2000). Every p-betting-game random oracle is also p-random; whether the two notions are equivalent is an open problem.

(1) We first show that $P^A \neq NP^A$ for every oracle A that is p-betting-game random.

Ideally, we would extend (1) to p-random oracles. We show that answering this either way would imply an unrelativized complexity class separation:

(2) If $P^A \neq NP^A$ relative to every p-random oracle A, then BPP $\neq$ EXP.

(3) If $P^A = NP^A$ relative to some p-random oracle A, then P $\neq$ PSPACE.

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 (1) to show that PH^A is infinite relative to oracles A that are p-betting-game random. Showing that PH^A separates at even its first level would also imply an unrelativized complexity class separation:

(4) If $NP^A \neq coNP^A$ for a p-betting-game measure 1 class of oracles A, then NP $\neq$ EXP.

(5) If PH^A is infinite relative to every p-random oracle A, then PH $\neq$ EXP.

We also consider random oracles for time versus space, for example:

(6) $L^A \neq P^A$ relative to every oracle A that is p-betting-game random.

1 Introduction

Bennett and Gill [5] initiated the study of random oracles in computational complexity, proving that $P^A \neq NP^A$ for a random oracle A, with probability 1. Subsequent work showed that this holds for *individual* random oracles. Book, Lutz, and Wagner [7] showed that $P^A \neq NP^A$ for every oracle A that is algorithmically random in the sense of Martin-Löf [19]. Lutz and Schmidt [18] improved this further to show $P^A \neq NP^A$ for every oracle A that is pspace-random [16].

*Department of Computer Science, University of Wyoming. jhitchco@cs.uwyo.edu

†Department of Mathematics, Computer Science & Physics, Roanoke College. sekoni@roanoke.edu

‡Department of Mathematics and Computer Science, Northern Michigan University. hshafei@nmu.edu

We investigate whether this extends to individual polynomial-time random oracles [16, 2]. 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 [1] 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 [5] that $\mathrm{P}^A = \mathrm{BPP}^A$ holds for a random oracle A, with probability 1. Allender and Strauss's proof relies on derandomization [22] and is a different approach than Bennett and Gill used. For P vs NP oracles, the best known is the pspace-randomness result of Lutz and Schmidt [18]. Because the class $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has Hausdorff dimension 1 [13], there is a fundamental limit to how strongly a martingale can succeed on the class [14].

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 3 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 [5], 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 that martingales are forced to bet on strings in lexicographic order. Buhrman et al. [8] 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 [8]. The adaptiveness in betting games allows us to simulate P^A algorithms. We show in Section 3 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 6, 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 [5] also showed that $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for a random oracle A, with probability 1. Rossman, Servedio, and Tan [23] answered a longtime open question [12] 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. In Section 4, we show that PH is infinite relative to space-bounded random oracles. 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.

In Section 5, we consider time versus space. Bennett and Gill [5] showed that $\mathrm{L}^A \neq \mathrm{P}^A$ and $\mathrm{PSPACE}^A \neq \mathrm{E}^A$ relative to a random oracle A. We show these separations also hold relative to p-betting game random oracles.

This paper is organized as follows. Section 2 contains the preliminaries, particularly on resource-bounded measure and betting games. Section 3 contains the results on random oracles for P vs. NP, as well as extensions to P vs. UP. The results on random oracles for PH are in Section 4. Random oracles for time vs. space are addressed in Section 5. We present limitations to improving our results in Section 6.

2 Preliminaries

We use standard notation. The binary alphabet is $\Sigma = \{0,1\}$, the set of all binary strings is Σ^*, the set of all binary strings of length n is Σ^n, and the set of all infinite binary sequences is Σ^∞. The empty string is denoted by λ. We use the standard enumeration of strings, $s_0 = \lambda, s_1 = 0, s_2 = 1, s_3 = 00, s_4 = 01, \ldots$, and the standard lexicographic ordering of strings corresponds to this enumeration. The characteristic sequence of a language A is the sequence $\chi_A \in \Sigma^\infty$, where $\chi_A[n] = 1 \iff s_n \in A$. We refer to $\chi_A[s_n] = \chi_A[n]$ as the characteristic bit of s_n in A. A language A can alternatively be seen as a subset of Σ^*, or as an element of Σ^∞ via identification with its characteristic sequence χ_A. Given strings x, y 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. $\lambda + 4 = 01$. Similarly we denote by $A[x, y]$ the substring of the characteristic sequence χ_A that corresponds to the characteristic bits of the strings in $[x, y]$.

2.1 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 [16, 17, 2, 8, 11].

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) [21, 20]. The concept was introduced to computational complexity by Buhrman et al. [8]. First, we recall the definition of a martingale:

Definition. A *martingale* is a function $d : \Sigma^* \to [0, \infty)$ such that for all $w \in \Sigma^*$, 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 [8].

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 G 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 the next section.

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 $X \subseteq S^\infty[G]$.

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.

By adding a resource bound Δ on the computation of a betting game or martingale, we get notions of resource-bounded measure on Σ^∞. For this paper the resource bounds we use are $\mathrm{p} = \mathrm{DTIMEF}(n^{O(1)})$, $\mathrm{p}_2 = \mathrm{DTIMEF}(2^{(\lg n)^{O(1)}})$, $\mathrm{p}_3 = \mathrm{DTIMEF}(2^{2^{(\lg \lg n)^{O(1)}}})$, and the analogous space bounds pspace, $\mathrm{p}_2$space, and $\mathrm{p}_3$space. We say a class $X \subseteq \Sigma^\infty$ 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 [16]. 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. [8] 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. Most importantly, no polynomial-time betting game can succeed on the class $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.

3 Random Oracles for P vs. NP

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

Theorem 3.1. *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 a concatenation of 2^n length n strings $w_1, w_2, \cdots, w_{2^n}$; i.e. $w = w_1 w_2 \cdots w_{2^n}$. $\mathrm{Tribes}_{2^n,n}(w)$ 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 $L_A \in \mathrm{NP}^A$. Now we define a betting game G that succeeds on the set $X = \{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$, 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 infinitely-many "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 querying 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 so far by G.

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 $0^{n_j} \notin L_A$. 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 $[0^{n_j}+1, 0^{n_j}+n_j 2^{n_j}]$, $\omega[i]$ is the status of string $0^{n_j}+i$, $\star$ indicates the string has not been queried by G yet, for queried strings, 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 $[0^{n_j}+1, 0^{n_j}+n_j 2^{n_j}]$ 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 $[0^{n_j}+1, 0^{n_j}+n_j 2^{n_j}]$ is revealed, a_l is then updated to $G_{a_l}(\omega)$. Given $\omega \in \{0,1,\star\}^{n_j 2^{n_j}}$ 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 $\omega \in \{0,1\}^{n_j 2^{n_j}}$ 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 $[0^{n_j}+1, 0^{n_j}+n_j 2^{n_j}]$ 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 unqueried 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. □

Next, we consider random oracles for P vs. NP relative to martingales rather than betting games. Note that Theorem 3.2 is incomparable with Theorem 3.1 because it's not known whether either of the two classes p-betting-game-random oracles and p_2-random oracles is a subset of the other class.

Theorem 3.2. $P^A \neq NP^A$ *for every* p_2*-random oracle* A*.*

Proof. The proof follows from a few observations from the proof of the previous theorem. We want to show that there is a p_2-martingale that succeeds on any oracle A such that, $P^A = NP^A$. In the previous theorem, we partition the set $X = \{A \mid P^A = NP^A\}$. Let us denote these partitions as X_1 and X_2, where X_1 and X_2 are the set of oracles for which $\{0^n \mid 0^n \notin L_A\}$ is finite and infinite, respectively. We designed a p-betting-game that succeeded on any oracle in X_1. Furthermore, we argued that the betting-game succeeds on X_1 no matter the order in which it bets on strings. Therefore, we can bet on the strings of such oracles in the standard ordering of strings. Thus, there is a p_2-measure 0 oracle that succeeds on X_1.

It remains to show that there is a p_2-martingale that succeeds on any $A \in X_2$. Our strategy for succeeding on a A is modification of the strategy of the betting-game used to succeed on X_2 in the previous theorem. But in this situation we are trying to succeed on a single oracle rather than an infinite set of oracles. Let M^A be a polynomial time oracle TM that decides L_A. In the betting-game of the previous theorem, we used the order of the polynomially many queries made by M^A on input 0^n to request some of the strings we bet on. Once $M^A(0^n)$ made all the queries, we could easily recognize an event that occurred infinitely often with probabilities that guaranteed success. We can achieve the same with a p_2-martingale. A p_2-martingale has enough time to recognize and compute the probability of the aforementioned event. Instead of requesting the strings queried by $M^A(0^n)$, the martingale tries all the possible responses to the polynomially many queries made by the TM. This can be done in p_2-time. The success of the martingale follows from essentially the same argument presented in the proof of previous theorem. □

Corollary 3.3. $\{A \mid P^A \neq NP^A\}$ *has* p_3*-measure 1.*

Proof. From Theorem 3.2, for each A with $P^A = NP^A$, there is a p_2-martingale that succeeds on A. The sum of all p_2-martingales is a p_3-martingale [16]. Therefore the class $\{A \mid P^A = NP^A\}$ has p_3-measure 0. □

Whether Corollary 3.3 can be improved to p_2-measure is an open problem that we address in Section 6.

3.1 Random Oracles for P vs. UP

We now show that Theorem 3.1 extends to oracles for P vs. UP.

Theorem 3.4. *The class* $\{A \mid P^A \neq UP^A\}$ *has* p*-betting-game measure 1. In particular,* $P^A \neq UP^A$ *for every* p*-betting-game random oracle* A*.*

Proof. The proof of this theorem is similar to Theorem 3.1. So we only focus on the main differences. For any oracle A we use the test language

$$L'_A = \{0^n \mid \text{Tribes}_{2^n, n+\lg n}(A[0^n + 1, 0^n + (n + \lg n)2^n]) = 1\}.$$

Note the change in the size of each tribe, from n used in Theorem 3.1 to $n+\lg n$ used here. Consider the set X of oracles A such that, $A[0^n+1, 0^n+(n+\lg n)2^n]$ has at most one tribe with all 1s for all but finitely many n. It is easy to see that $L'_A \in \mathrm{UP}^A$ for any oracle $A \in X$. A UP machine simply has to guess the address of the unique tribe and then verify the membership of $n+\lg n$ strings. We will first show that the set X^{c} of oracles A such that $A[0^n+1, 0^n+(n+\lg n)2^n]$ has more than one tribe with all 1s infinitely often, has p-measure zero. In particular we can succeed on X^{c} no matter the order in which a p-betting game queries strings. We also describe a p-betting game that succeeds on X. Therefore, we will be able to combine these two p-betting games just like we did in Theorem 3.1 and the theorem will follow.

Succeeding on X^{c}: Our betting game has initial capital $\sum_{n=0}^{\infty} 1/n^{1.5}$. For each n, the betting game reserves $1/n^{1.5}$ to bet on the strings in the range $[0^n+1, 0^n+(n+\lg n)2^n]$. As before, we view these strings as tribes, each tribe consisting of $n+\lg n$ strings. There are fewer than 2^{2n} pairs of tribes. Our betting game simply bets on every possible pair of tribes being contained in A, the oracle being bet on. We bet on each pair of tribes with capital $1/n^{1.5}2^{2n}$. For every correct guess, the capital reserved for the pair grows from $1/n^{1.5}2^{2n}$ to $2^{2(n+\lg n)}/n^{1.5}2^{2n} = \Omega(n^{0.5})$. Since any oracle in X^{c} will have its capital increase by $\Omega(n^{0.5})$ infinitely often, it follows that our betting game succeeds on X^{c}. It is easy to see that this betting game can be implemented in p and also doesn't depend on the order in which strings are queried.

Succeeding on $X \cap \{A \mid \mathrm{P}^A = \mathrm{UP}^A\}$: Let A be in this class. There is some polynomial time oracle TM M_i^A that decides L'_A. To complete the proof, we need to design a p-betting-game that succeeds on the intersection. The design of the betting game is almost identical to that of Theorem 3.1. □

Analogously, Theorem 3.2 extends to P vs. UP:

Corollary 3.5. $\mathrm{P}^A \neq \mathrm{UP}^A$ *for every* p_2*-random oracle* A.

4 Betting Game Random Oracles for the Polynomial-Time Hierarchy

Rossman et al. [23] defined Sipser_d^n, a family of n-variable read-once monotone depth-d circuits for which the following theorem holds.

Theorem 4.1 (Rossman, Servedio, and Tan [23])**.** *Let* $2 \leq d \leq \frac{c\sqrt{\log n}}{\log\log n}$*, where* $c > 0$ *is an absolute constant. Then any circuit* C *of depth at most* $d-1$ *and size at most* $2^{n^{\frac{1}{6(d-1)}}}$ *over* $\{0,1\}^n$ *agrees with* Sipser_d^n *on at most* $2^n(\frac{1}{2}+n^{-\omega(1/d)})$ *inputs.*

This theorem implies that the polynomial hierarchy is infinite relative to a random oracle [12, 23]. We strengthen this result to show that the polynomial hierarchy is infinite relative to any $\mathrm{p}_2$space random oracle.

The following lemma is a combination of two lemmas from [12].

Lemma 4.2 (Håstad [12])**.** *Let* M^A *be a* $\sum_k^{\mathrm{P},A}$ *oracle TM which runs in time* n^c *on input* x *of length* n*, and* $\mathcal{Q}_n(M)$ *be the set of boolean variables that represent answers to queries that can be made by* M *on input* 0^n*. Then there is a depth* $k+2$ *circuit* C *of size* 2^{n^c} *with input set* $\mathcal{Q}_n(M)$ *that computes* $M^A(0^n)$ *for any oracle* A*.*

Lemma 4.3. *For any oracle A, and its test language L_A, if the following conditions hold, then A is not* $\mathrm{p}_2$space *random.*

1. *The membership of 0^n in L_A depends on the membership of the strings of length at most n^k.*
2. *L_A is decided by an oracle TM M that queries only strings of length at most n^k.*
3. *For any string w of length at most 2^{n^k+1}, the conditional probability $Pr(B|w)$ of $M^B(0^n) = L_B[0^n]$ given that $w \sqsubseteq B$ is computable in $O(2^{n^k})$ space, where $B \in \{0,1\}^\infty$ is a randomly selected oracle with prefix string w.*
4. *For some constant $\epsilon > 0$, $Pr(B|A[\lambda, 0^n]) \leq 1 - \epsilon$ for all but finitely many n.*

Proof. We will show any oracle A as described in the statement of the theorem is not p_2-random by defining a $\mathrm{p}_2$space martingale that succeeds on A. Consider the following sequence of integers,

$$n_j = \begin{cases} 2, & j = 0 \\ (n_{j-1}+1)^k, & j > 0. \end{cases}$$

For sufficiently large j, M^B cannot query any string of length n_j when running on input $0^{n_{j-1}}$. Also, the membership of $0^{n_{j-1}}$ does not depend on the membership of any string of length greather than n_j. We now define our martingale d.

$$d(w) = \begin{cases} 1 & , |w| \leq 2^{c+1} - 1 \\ \frac{d(w[\lambda, 0^{n_j-1}])}{\Pr(B|w[\lambda,0^{n_j-1}])}\Pr(B|w) & , 2^{c+1} \leq 2^{n_{j-1}} < |w| \leq 2^{n_j+1} - 1 \end{cases}.$$

The constant c is any n_j such that $\Pr(B|A[\lambda, 0^n]) \leq 1 - \epsilon$ for all $n > c$. It is easy to verify that d is a martingale. In fact, by the third condition in the statement of the theorem, d is $\mathrm{p}_2$space computable. We now argue that d succeeds on any oracle A that satisfies the statement of the theorem. Given such an oracle A and $n_j > c$ then,

$$\begin{aligned} d(A[\lambda, 0^{n_j}]) &= \frac{d(A[\lambda, 0^{n_j-1}])}{\Pr(B|A[\lambda, 0^{n_j-1}])}\Pr(B|A[\lambda, 0^{n_j}]) \\ &\leq \frac{d(A[\lambda, 0^{n_j-1}])}{1-\epsilon}. \end{aligned}$$

$\Pr(B|A[\lambda, 0^{n_j}]) = 1$ because, $M^B(0^{n_j-1}) = L_B[0^{n_j-1}]$ for every random oracle B such that $A[\lambda, 0^{n_j}]) \sqsubseteq B$. This follows because, $M^A(0^{n_j-1}) = L_A[0^{n_j-1}]$, M can only query strings of length less than n_j on input 0^{n_j-1}, and $L_B[0^{n_j-1}]$ only depends on the membership of strings of length less than n_j. Consequently, the random bits of any oracle B with $A[\lambda, 0^{n_j}] \sqsubseteq B$ do not affect either $M^B(0^{n_j})$ or $L_B[0^{n_j}]$. If we apply the same argument to $d(A[\lambda, 0^{n_j-1}])$, we see that d succeeds on A. □

Theorem 4.4. *For any $k \in \mathbb{N}$, $\sum_k^{\mathrm{P},A} \neq \sum_{k+1}^{\mathrm{P},A}$ for every* $\mathrm{p}_2$space*-random oracle A.*

Proof. The result will follow by applying Lemma 4.3. We show that there is a $\mathrm{p}_2$space betting game that succeeds on any oracle A such that $\sum_k^{\mathrm{P},A} = \sum_{k+1}^{\mathrm{P},A}$. Given such an A we define a test language

$$L_A = \{0^n | \mathrm{Sipser}_{k+3}^n(A[0^n + 1, 0^n + N_n]) = 1\}.$$

Where, $N_n = \Theta((n2^n)^{k+2})$ is the number of variables of Sipser^n_{k+3}. We do not present the complete definition of Sipser^n_{k+3} because of its complexity. Implicit in its definition in [23], is the fact that $L_A \in \sum^{\text{P},A}_{k+3}$. It easy to see that the first two conditions of Lemma 4.3 are now satisfied.

We now show that the last two conditions are also satisfied. Since $\sum^{\text{P},A}_k = \sum^{\text{P},A}_{k+1}$, it follows that $L_A \in \sum^{\text{P},A}_k$. Let M^A be a $\sum^{\text{P},A}_k$ oracle TM that decides L_A. By Lemma 4.3 there is a depth $k+2$ circuit of size 2^{n^c} that computes Sipser^n_{k+3}. Theorem 4.1 tells us that the probability of a this circuit agreeing with Sipser^n_{k+3} is at most $2/3$ for all but finitely many n. Therefore the probability of $M^B(0^n) = L_B[0^n]$ for a randomly selected oracle is at most $2/3$ for all but finitely many n. In fact, for any string w ($|w| < 2^{n^k}$) we can easily compute, in $O(2^{n^k})$ space, the probability of $M^B(0^n) = L_B[0^n]$ where $w \sqsubseteq B$ is a randomly selected oracle. This is done by a brute force search which only requires $O(2^{n^k})$ space. These facts show that the last two conditions of the Lemma 4.3 also hold. Therefore it follows that A is not p_2-random. □

Corollary 4.5. $\{A \mid \text{PH}^A$ *is infinite*$\}$ *has* $\text{p}_3$space *measure 1.*

5 Random Oracles for Time vs Space

In this section we deal with space bounded TMs. We follow the convention of not counting the space used by queries to the write-only tape of the space-bounded TM.

Theorem 5.1. *The class* $\{A \mid \text{L}^A \neq \text{P}^A\}$ *has* p*-betting-game measure 1. In particular,* $\text{L}^A \neq \text{P}^A$ *for every* p*-betting-game random oracle* A*.*

Proof. For any oracle A we define the test language L_A as follows:

$$L_A = \{x | A(x1)A(x10)\cdots A(x10^{|x|-1}) \in A\}$$

i.e. $x \in L_A$ if and only if the string $x[A] = A(x1)A(x10)\cdots A(x10^{|x|-1})$ belongs to A. It is easy to see that $L_A \in \text{P}^A$ since any length n string can be decided with $n+1$ queries to the oracle. We now describe a p-betting game that succeeds on $X = \{A \mid \text{L}^A = \text{P}^A\}$. Let $M_1, M_2, \cdots$ be an enumeration of all logspace TMs such that M_i uses at most $\lg i \lg n$ space. A $\lg i \lg n$ space-bounded oracle TM on input 0^n has $n^{O(\lg i)}$ configurations and thus can query at most $n^{O(\lg i)}$ strings. We denote by $\mathcal{Q}_n(M_i)$ the set of strings that can be queried by M_i when running on input 0^n.

In order to succeed on every $A \in X$ our betting game goes through the list of logspace TMs until it finds one that decides L_A on some tally set. At every stage of betting our betting game is directed by some M_i. Two cases arise while simulating M_i^A on input 0^n. Either $0^n[A]$ is queried or it isn't. If $0^n[A]$ is queried then we use its bits to predict the membership of the strings $0^n1, 0^n10, \cdots, 0^n10^{n-1}$, otherwise we use the output of $M_i^A(0^n)$ to predict the membership of $0^n[A]$. Details follow.

The betting game operates in stages, similar to the betting game of Theorem 3.1. So we omit some of the minor details. The betting game starts with initial capital 2. It reserves $a_i = b_i = 1/i^2$ of its initial capital to bet with TM M_i.

At the beginning of stage j, the betting game selects a logspace TM M_i, where i is the smallest index of a TM that hasn't made a mistake. What it means for a TM to make a mistake will be specified shortly. The betting game then computes $\mathcal{Q}_n(M_i)$, where n is the smallest integer that is greater than the length of all the strings queried in the previous stage. Recall that $\mathcal{Q}_n(M_i)$ is the set of the $n^{O(\lg i)}$ strings that can be queried by M_i on input 0^n. The betting game then simulates

M_i on input 0^n and answers any query by either looking up already queried strings or requesting unqueried strings. The strings we are interested in betting on are $0^n1, 0^n10, \cdots, 0^n10^{n-1}$ and $0^n[A]$. The portion a_n is used to bet on $0^n1, 0^n10, \cdots, 0^n10^{n-1}$ while the b_i portion is used to bet on $0^n[A]$.

Betting with a_n: the betting game splits a_n equally among the length n strings in $\mathcal{Q}_n(M_i)$. We use each $w \in \mathcal{Q}_n(M_i)$ to predict the membership of the n strings
$0^n1, 0^n10, \cdots, 0^n10^{n-1}$ i.e., the ith bit of each w is used to predict the membership of 0^n10^{i-1}. The entire portion of a_n reserved for betting with each w is used to bet on each prediction of w. This doubles the portion reserved for each w by a factor of two for every correct prediction. If $\mathcal{Q}_n(M_i)$ contains

$$0^n[A] = A(0^n1)A(0^n10)\cdots A(0^n10^{n-1}),$$

the betting-game makes n correct predictions when betting with $0^n[A]$. Therefore, the betting-game's capital reserved for betting on $0^n[A]$ grows from

$$1/n^2 2^{O(\lg i)\lg n}$$

to

$$\Omega(2^n/n^2 2^{\lg^2 n})$$

by the end of this stage.

Betting with b_i: once the betting game is done simulating M_i on 0^n and requesting the strings $0^n1, 0^n10, \cdots, 0^n10^{n-1}$, if $0^n[A]$ wasn't queried, then the betting game bets with b_i that the characteristic bit of $0^n[A]$ is $M_i(0^n)$. We say M_i makes a mistake if $M_i^A(0^n) \neq A(0^n[A])$. If M_i makes a mistake then b_i becomes 0, otherwise it doubles.

To round up this stage of betting, the betting game requests all unqueried strings of length up to the length of the longest string queried by the betting game so far. It is easy to see that this betting game can be implemented in p. The longest step is computing the set $\mathcal{Q}_n(M_i)$ of all strings that can be queried. This can be done by examining the $O(2^{\lg^2 n}) = o(2^n)$ configurations of any M_i on input 0^n, where $i \leq n$.

Finally we argue that the betting game succeeds on any $A \in X$. For such an oracle A the betting game must eventually select some TM $M_{i^\star}$ that correctly decides the test language L_A on all the inputs we run it on. Therefore, for every stage of betting, $M_{i^\star}$ either queries $0^n[A]$ or it doesn't. When it does then a_n grows to $\Omega(2^n/n^2 2^{\lg^2 n})$, when it doesn't then $b_i^\star$ doubles. The betting game succeeds because, $b_i^\star$ never decreases and a_k grows to $\Omega(2^n/n^2 2^{\lg n})$ whenever $b_i^\star$ doesn't increase. □

Theorem 5.2. $\mathrm{L}^A \neq \mathrm{P}^A$ *for every* p_2*-random oracle* A*.*

Proof. This proof is similar to that of Theorem 5.1. We need to design a martingale, rather than a betting game, that succeeds on any oracle A such that $\mathrm{L}^A = \mathrm{P}^A$. The martingale bets the same way as the betting game in the previous theorem, the difference being the order in which strings are queried. In order to convert the betting game into a martingale, we try all possible query answers in parallel instead of actually querying the desired string. This way, we can query strings in the standard ordering. Details follow.

Let X and L_A be as defined in Theorem 5.1. Given $A \in X$, let M_i be a logspace TM that decides L_A in $\lg i \lg n$ space. Recall that $\mathcal{Q}_n(M_i)$ is the set of $n^{O(\lg i)}$ strings that can be queried by M_i when it is run on input 0^n. Two cases arise, either $0^n[A] = A(0^n1)A(0^n10)\cdots A(0^n10^{n-1}) \in \mathcal{Q}_n(M_i)$ infinitely often, or $0^n[A] \notin \mathcal{Q}_n(M_i)$ for all but finitely many n.

In the first case we describe a martingale that succeeds any oracle A such that, $0^n[A] \in \mathcal{Q}_n(M_i)$ infinitely often. Similar to Theorem 5.1, the martingale bets in stages and breaks its initial capital into infinite portions $a_j = 1/j^2$, with some a_k used for betting in stage j. At the beginning of stage j the martingale computes $\mathcal{Q}_n(M_i)$, where n is the length of the smallest integer greater than all the strings seen in the previous stage. The martingale then splits a_n equally among all the length n strings in $\mathcal{Q}_n(M_i)$. Each of these length n strings along with its portion of a_n is used to predict the membership of the strings $0^n1, 0^n10, \cdots, 0^n10^{n-1}$. These are the only strings the martingale bets on. To complete stage j the martingale queries all the remaining strings of length at most $2n = |0^n10^{n-1}|$. It is easy to see that whenever $0^n[A] \in \mathcal{Q}_n(M_i)$, the capital reserved for betting on some length n string in $\mathcal{Q}_n(M_i)$ grows from at least $1/n^2 2^{\lg^2 n}$ to $2^n/n^2 2^{\lg^2 n} = \omega(1)$. Therefore our martingale succeeds on all oracles A such that $0^n[A] \in \mathcal{Q}_n(M_i)$ infinitely often.

In the second case we design a martingale that succeeds on any oracle A such that, $0^n[A] \notin \mathcal{Q}_n(M_i)$ for all but finitely many n. This martingale also bets on strings in stages. It starts betting on strings of length n_0, which is the length at which $0^n[A] \notin \mathcal{Q}_n(M_i)$ holds for all $n \geq n_0$. The martingale starts with initial capital 1 and operates in each stage as follows. Let $\mathcal{Q}_{\geq n}(M_i)$ denote the set of strings in $\mathcal{Q}_n(M_i)$ with length at least n . At the beginning of stage j, the martingale simulates M_i on input 0^n for each of the $O(2^{n^{\lg i}+n})$ subsets of $\mathcal{Q}_{\geq n}(M_i) \cup \{0^n1, 0^n10, \cdots 0^n10^{n-1}\}$. The martingale's current capital c_j is split up equally among each subset and $0^n[A]$ is also computed for each subset. For each subset, whenever the martingale queries a string in $s \in \mathcal{Q}_{\geq n}(M_i) \cup \{0^n1, 0^n10, \cdots 0^n10^{n-1}\}$, the martingale bets the entire capital reserved for the subset according to the membership of s in the subset. Also, the martingale bets on the string $0^n[A]$ associated with each subset. For each subset the martingale computes $0^n[A]$ and simulates M_i on 0^n with queries answered according to the subset. The martingale then bets the entire capital reserved for each subset on $M_i(0^n)$ being the characteristic bit of $0^n[A]$. The stage ends by querying all the remaining strings of length at most the length of the longest string queried so far.

Clearly this martingale can be implemented in time polynomial in 2^n. We now show that this martingale succeeds on any oracle A such that $0^n[A] \notin \mathcal{Q}_n(M_i)$ holds for all $n \geq n_0$. In each stage j, the capital at the beginning of the stage c_j doubles. This is because c_j is divided into an equal portion for every subset of $\mathcal{Q}_{\geq n}(M_i) \cup \{0^n1, 0^n10, \cdots 0^n10^{n-1}\}$. Since there is only one subset that is consistent with the oracle A we are betting on, the share reserved for this subset will grow back to c_j after betting on the strings in $\mathcal{Q}_{\geq n}(M_i) \cup \{0^n1, 0^n10, \cdots 0^n10^{n-1}\}$. Finally, the martingale will bet according to the output $M_i(0^n)$, on the string $0^n[A]$ associated with this subset. Since M_i^A decides L_A and $0^n[A] \notin \mathcal{Q}_n(M_i)$ for all sufficiently large inputs, the output of the TM is correct, therefore the current capital reserved for the correct subset is doubled. Therefore, our martingale's capital is doubled after every stage of betting, thus it succeeds on A. □

Corollary 5.3. $\{A \mid \mathrm{L}^A \neq \mathrm{P}^A\}$ *has* p_3*-measure 1.*

Using a result of Book [6], we extend Theorems 5.1 and 5.2 to random oracles for PSPACE vs E.

6 Limitations

In this section we examine the possibility of extending Theorem 3.1. 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 3.1 cannot

be improved to p-random oracles would separate PSPACE from P. We also consider limitations of our other results.

6.1 Does $P^A \neq NP^A$ for every p-random oracle A?

We showed in Theorem 3.1 that $P^A \neq NP^A$ 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 [8]. This is based on the following theorem and the result that $\leq_T^P$-complete languages for EXP have p-betting-game measure 0 [8].

Theorem 6.1 (Buhrman et al. [8]). *If the class of $\leq_T^P$-complete languages for* EXP *has* p_2*-measure zero then* BPP $\neq$ EXP*.*

We show that improving Theorem 3.1 to p-random oracles would also imply BPP $\neq$ EXP. First, we prove the following for p_2-measure.

Theorem 6.2. *If $\{A \mid P^A \neq NP^A\}$ has p_2-measure 1, then* BPP $\neq$ EXP*.*

Proof. If L is any $\leq_T^P$-complete language for EXP, then

$$NP^L \subseteq EXP \subseteq P^L \subseteq NP^L.$$

Therefore the class of $\leq_T^P$-complete languages for EXP is a subset of $\{A \mid P^A = NP^A\}$. If $\{A \mid P^A = NP^A\}$ has p_2-measure 0 then so does the class of $\leq_T^P$-complete languages of EXP. Theorem 6.1 implies that BPP $\neq$ EXP. □

We have the following for p-random oracles by the universality of p_2-measure for p-measure [16].

Corollary 6.3. *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 [16]: $S^\infty[d] \subseteq S^\infty[d']$ for every p-martingale d. Then d' succeeds on $\{A \mid P^A = NP^A\}$. □

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

Given Theorem 6.2, we consider the possibility of whether $\{A \mid P^A = NP^A\}$ does not have p-measure 0. Because Lutz and Schmidt [18] 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 6.4 (Lutz and Schmidt [18]). *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 [8], Theorem 6.4 follows as a corollary to Theorem 3.1.

Lemma 6.5. *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 [15] and its output is in $\{0,1\}^{\leq p(n)}$, for some polynomial p. Consider the language $L_d = \{\langle w, i, b\rangle \mid$ the ith bit of $d(w)$ is $b\}$. Clearly $L_d \in$ PSPACE and hence also in P by our hypothesis. We can therefore compute $d(w)$ in polynomial time using L_d. □

Theorem 6.6. *If* $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ *does not have* p*-measure 0, then* $\mathrm{P} \neq \mathrm{PSPACE}$.

Proof. Assume $\mathrm{P} = \mathrm{PSPACE}$. Theorem 6.4 and Lemma 6.5 imply that $\{A \mid \mathrm{P}^A = \mathrm{NP}^A\}$ has p-measure 0. □

Corollary 6.7. *If there is a* p*-random oracle* A *such that* $\mathrm{P}^A = \mathrm{NP}^A$*, then* $\mathrm{P} \neq \mathrm{PSPACE}$.

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

Bennett and Gill [5] 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 [23] showed that PH^A is infinite relative to a random oracle, with probability 1.

Can we improve Theorems 3.1 and 4.4 to show that PH^A does not collapse for a p-betting-game random oracle? This also has complexity class separation consequences:

Theorem 6.8. *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:

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

Therefore $\sum_k^{\mathrm{P},A} = \prod_k^{\mathrm{P},A}$, which in turn implies that $\mathrm{EXP} \subseteq X_k$. Since EXP does not have p_2-betting-game measure zero [8] then neither does X_k. Hence, the Theorem follows. □

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

Corollary 6.9. *If* $\{A \mid \mathrm{NP}^A \neq \mathrm{coNP}^A\}$ *has* p_2*-betting-game measure 1, then* $\mathrm{NP} \neq \mathrm{EXP}$.

Because it is open whether betting games have a union lemma [8], it is not clear whether Corollary 6.9 may be extended to show that if $\mathrm{NP}^A \neq \mathrm{coNP}^A$ for every p-betting-game random oracle A, then $\mathrm{NP} \neq \mathrm{EXP}$. 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 6.10. *If* $\mathrm{NP}^A \neq \mathrm{coNP}^A$ *for every* p*-random oracle* A*, then* $\mathrm{NP} \neq \mathrm{EXP}$.

Corollary 6.11. *If* PH^A *is infinite for every* p*-random oracle* A*, then* $\mathrm{PH} \neq \mathrm{EXP}$.

6.4 Further Limitations

The technique used in Section 6.3 may be used to establish additional limitation results for other classes. In the next lemma, we abstract the argument used in Theorem 6.8.

Lemma 6.12. *Let* $\mathcal{C}, \mathcal{D} \subseteq \mathrm{EXP}$ *be relativizable classes such that* $\mathcal{C}^A \subseteq \mathrm{EXP}$ *for every* $A \in \mathrm{EXP}$*. Let*

$$X = \{A \mid \mathcal{C}^A \subseteq \mathcal{D}^A\}.$$

If $\mathrm{EXP} \not\subseteq X$*, then* $\mathcal{D} \neq \mathrm{EXP}$*. In particular, if* X *has* p_2*-betting-game measure 0, then* $\mathcal{D} \neq \mathrm{EXP}$.

Proof. Assume $\mathcal{D} = \mathrm{EXP}$. Let $A \in \mathrm{EXP}$. Then

$$\mathcal{C}^A \subseteq \mathrm{EXP} = \mathcal{D} \subseteq \mathcal{D}^A.$$

Therefore $\mathrm{EXP} \subseteq X$, so X doesn't have p_2-betting-game measure 0. □

Corollary 6.13. *1. If* $\{A \mid \mathrm{NP}^A \neq \mathrm{coNP}^A\}$ *has* p_2*-betting game measure 1, then* $\mathrm{NP} \neq \mathrm{EXP}$.

2. If $\{A \mid \mathrm{PH}^A$ *is infinite*$\}$ *has* p_2*-betting game measure 1, then* $\Sigma_k^{\mathrm{P}} \neq \mathrm{EXP}$ *for every* k.

3. If $\{A \mid \mathrm{PH}^A \neq \mathrm{PSPACE}^A\}$ *has* p_2*-betting game measure 1, then* $\mathrm{PH} \neq \mathrm{EXP}$.

4. If $\{A \mid \mathrm{IP}^A \neq \mathrm{PSPACE}^A\}$ *has* p_2*-betting game measure 1, then* $\mathrm{IP} \neq \mathrm{EXP}$.

5. If $\{A \mid \mathrm{UP}^A \neq \mathrm{NP}^A\}$ *has* p_2*-betting game measure 1, then* $\mathrm{UP} \neq \mathrm{EXP}$.

6. If $\{A \mid \mathrm{NP}^A \not\subseteq \mathrm{BPP}^A\}$ *has* p_2*-betting game measure 1, then* $\mathrm{BPP} \neq \mathrm{EXP}$.

Proof. 1. Apply the lemma with $\mathcal{C} = \mathrm{coNP}$ and $\mathcal{D} = \mathrm{NP}$.

2. Apply the lemma with $\mathcal{C} = \Sigma_{k+1}^{\mathrm{P}}$ and $\mathcal{D} = \Sigma_k^{\mathrm{P}}$.

3. Apply the lemma with $\mathcal{C} = \mathrm{PSPACE}$ and $\mathcal{D} = \mathrm{PH}$.

4. Apply the lemma with $\mathcal{C} = \mathrm{PSPACE}$ and $\mathcal{D} = \mathrm{IP}$.

5. Apply the lemma with $\mathcal{C} = \mathrm{NP}$ and $\mathcal{D} = \mathrm{UP}$.

6. Apply the lemma with $\mathcal{C} = \mathrm{NP}$ and $\mathcal{D} = \mathrm{BPP}$.

□

All of the classes in Corollary 6.13 are known to have classical measure 1 [5, 23, 3, 9, 10, 4]. Improving these results to betting game measure (or to resource-bounded measure) would have significant consequences for separating complexity classes.

7 Conclusion

We have shown that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every p-betting-game random oracle A (Theorem 3.1). Establishing whether this also holds for p-random oracles would imply either $\mathrm{BPP} \neq \mathrm{EXP}$ (Corollary 6.3) or $\mathrm{P} \neq \mathrm{PSPACE}$ (Corollary 6.7). These results, together with Theorems 4.4, 6.4, and 6.8, 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 $\{A \mid \mathrm{PH}^A$ collapses$\}$ have pspace-measure 0?

References

[1] 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.

[2] 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.

[3] L. Babai. Random oracles separate PSPACE from the polynomial-time hierarchy. *Information Processing Letters*, 26:51–53, 1987.

[4] R. Beigel. On the relativized power of additional accepting paths. In *Proceedings of the Fourth Annual Structure in Complexity Theory Conference*, pages 216–224. IEEE Computer Society, 1989.

[5] 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.

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

[7] 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.

[8] 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.

[9] J. Cai. With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy. *Journal of Computer and System Sciences*, 38:68–85, 1989.

[10] R. Chang, B. Chor, O. Goldreich, J. Hartmanis, J. Håstad, D. Ranjan, and R. Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

[11] 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.

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

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

[14] J. M. Hitchcock, J. H. Lutz, and E. Mayordomo. The fractal geometry of complexity classes. *SIGACT News*, 36(3):24–38, September 2005.

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

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

[17] 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.

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

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

[20] 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.

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

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

[23] B. Rossman, R. A. Servedio, and L.-Y. Tan. An average-case depth hierarchy theorem for Boolean circuits. In *Proceedings of the 56th Symposium on Foundations of Computer Science*, pages 1030–1048. IEEE Computer Society, 2015.