COMPLEXITY AND RAMSEY LARGENESS OF SETS OF ORACLES SEPARATING COMPLEXITY CLASSES

ALEX CREINER AND STEPHEN JACKSON

Abstract. We prove two sets of results concerning computational complexity classes. The first concerns a variation of the random oracle hypothesis posed by Bennett and Gill after they showed that relative to a randomly chosen oracle, $\boldsymbol{P} \neq \boldsymbol{NP}$ with probability 1. This hypothesis was quickly disproven in several ways, most famously in 1992 with the result that $\boldsymbol{IP} = \boldsymbol{PSPACE}$, in spite of the classes being shown unequal with probability 1. Here we propose a variation of what it means to be "large" using the Ellentuck topology. In this new context, we demonstrate that the set of oracles separating $\boldsymbol{NP}$ and $\boldsymbol{coNP}$ is not small, and obtain similar results for the separation of $\boldsymbol{PSPACE}$ from $\boldsymbol{PH}$ along with the separation of $\boldsymbol{NP}$ from $\boldsymbol{BQP}$. We demonstrate that this version of the hypothesis turns it into a sufficient condition for unrelativized relationships, at least in the three cases considered here. Second, we example the descriptive complexity of the classes of oracles providing the separations for these various classes, and determine their exact placement in the Borel hierarchy.

1. Introduction

In 1975, a paper by Baker, Gill and Solovay [11] demonstrated the subtlety of the $\boldsymbol{P}$ versus $\boldsymbol{NP}$ question by demonstrating the existence of oracles relative to which $\boldsymbol{P} \neq \boldsymbol{NP}$, as well as oracles relative to which $\boldsymbol{P} = \boldsymbol{NP}$. Subsequently Bennett and Gill in 1981 [2] demonstrated that if these classes were relativized to an oracle drawn at random (i.e., one in which every string is either in or out with probability $\frac{1}{2}$) then $\boldsymbol{P} \neq \boldsymbol{NP}$ with probability 1. They saw this as potentially strong evidence that $\boldsymbol{P}$ and $\boldsymbol{NP}$ were not equal in our reality. Upon showing similar results for other pairs of complexity classes, they closed the paper by framing the random oracle hypothesis: that if two classes demonstrated a certain relationship with probability 1 in the sense of drawing an oracle at random, then that relationship ought to hold in reality.

As early as 1983 [9], this hypothesis had been shown to be false in a variety of ways, most strikingly by Chang in 1994 who demonstrated that $\boldsymbol{IP} \neq \boldsymbol{PSPACE}$ with probability 1[4], despite Shamir's result just two years earlier proving that $\boldsymbol{IP} = \boldsymbol{PSPACE}$ unrelativized [10]. In their original paper, Bennet and Gill anticipated that their hypothesis was likely false, and that the condition might have to be strengthened. They wrote:

> We believe that this hypothesis, *or a similar but stronger one*, captures a basic intuition of the pseudorandomness of nature from which many apparently true complexity results follow.[Our emphasis]

We propose an alternative hypothesis, involving a different notion of "largeness" which seems to work in the cases we consider in the sense that if two classes are separated relative to a large set of oracles, then the unrelativized classes are distinct. Furthermore, the set of oracles relative to which the complexity classes we consider are separated is not small. We note that other notions of largeness have also been considered by various authors. For example, it was shown (see for example [8]) that these separating sets of oracles were large in the category sense, that is, they are comeager in the standard topology on the Cantor space 2^ω. The notion we consider is the notion of comeagerness with respect to the Ellentuck topology (defined in §3) on 2^ω, identified with $\mathcal{P}(\omega)$ or equivalently $[\omega]^\omega$, the set of increasing functions from ω to ω (we ignore finite sets here, as they correspond to trivial oracles). The Ellentuck topology arises naturally in logic and set theory, and been extensively studied. It is closely related to a forcing notion called Mathias forcing, and this provides a sometimes convenient alternate point of view. The Ellentuck topology, or Mathias forcing, naturally arises in studying infinitary partition properties on ω. The classical Ramsey theorem says that ω satisfies the finite exponent partition property, that is $\omega \to (\omega)^k_\ell$ for any $k, \ell \in \omega$, in the Erdös-Rado notation (see §3 for a more complete statement). Assuming the axiom of choice, AC, ω cannot satisfy an infinite exponent partition relation, however it can hold for certain classes of definable sets. The Galvin-Prikry theorem asserts that for Borel subsets $A \subseteq [\omega]^\omega$, the infinite exponent partition property holds (again, see §3 for complete statements). This was extended by Silver to all analytic ($\mathbf{\Sigma}^1_1$) sets. Mathias introduced a forcing notion which could give these results, and Ellentuck reformulated this into a topological notion, the Ellentuck topology. Sets $A \subseteq [\omega]^\omega$ for which a strong form of the partition property holds (the completely Ramsey sets, see §3) correspond to sets which are topologically regular, that is have the Baire property, with respect to this topology. The notion of measure, category, and largeness with respect to category in the Ellentuck topology, which we call *Ramsey large*, are thus natural notions to use in order to measure the size of sets of oracles. One of the goals of this paper is to investigate the sets of oracles providing separations between the various computational complexity classes with respect to the notion of Ramsey largeness.

In Theorem 3.3 and Corollary 3.4 we show that the set $\boldsymbol{O}$ of oracles separating $\boldsymbol{P}$ and $\boldsymbol{NP}$, and the set $\boldsymbol{O'}$ of oracles relative to which $\boldsymbol{NP} \neq \boldsymbol{co\text{-}NP}$ are both Ramsey positive (that is, they are not Ramsey small). This result agrees with the measure and category versions, however unlike those cases we are not able to show these sets are actually Ramsey large. Moreover, in Theorem 3.6 we show that if $\boldsymbol{O}$ is Ramsey large then $\boldsymbol{P} \neq \boldsymbol{NP}$. Thus, the version of the random oracle hypothesis using Ramsey largeness instead of measure holds for the class $\boldsymbol{O}$. In Theorem 3.5 we consider quantum computational complexity classes, and show that the oracles relative to which $\boldsymbol{NP} \not\subseteq \boldsymbol{BQP}$ is also Ramsey positive. In Theorem 3.7 we obtain the corresponding version of the random oracle hypothesis, that is, we show that if this set is Ramsey large, then $\boldsymbol{NP} \not\subseteq \boldsymbol{BQP}$. In Theorem 3.9 we show that the situation for the classes $\boldsymbol{PSPACE}$ and $\boldsymbol{PH}$ is somewhat different, Namely, we show that if the set of oracles separating these two classes is Ramsey positive, then $\boldsymbol{PSPACE} \neq \boldsymbol{PH}$. It thus remains an open question whether this set is Ramsey positive, however this result shows that the Ramsey version of the random oracle hypothesis holds for these classes as well. We conclude the section by discussing

what this survey reveals about the "Ramsey oracle hypothesis" being true more generally.

A second goal of the paper is consider the descriptive complexity of the various sets of oracles separating these computational complexity classes. Here the notions of descriptive set theory provide a standard way of calculating the complexity of a set of reals (oracles). Recall that in any topological space X the collection of Borel sets is defined as the smallest σ-algebra containing the open sets. This is stratified into a hierarchy, the Borel hierarchy, as follows. We let $\mathbf{\Sigma}^0_1$ denote the open sets, $\mathbf{\Pi}^0_1$ the closed sets, and $\mathbf{\Delta}^0_1 = \mathbf{\Sigma}^0_1 \cap \mathbf{\Pi}^0_1$ the clopen sets. In general, for $\alpha < \omega_1$ a countable ordinal we let $\mathbf{\Sigma}^0_\alpha$ be the sets A which can be written as a countable union $A = \bigcup_{n\in\omega} A_n$ where each $A_n \in \mathbf{\Pi}^0_{\alpha_n}$ for some $\alpha_n < \alpha$. Similarly, $A \in \mathbf{\Pi}^0_\alpha$ if $A = \bigcap_{n\in\omega} A_n$ where each $A_n \in \mathbf{\Sigma}^0_{\alpha_n}$ for some $\alpha_n < \alpha$. Also, we let $\mathbf{\Delta}^0_\alpha = \mathbf{\Sigma}^0_\alpha \cap \mathbf{\Pi}^0_\alpha$. It is a standard fact that in any uncountable Polish space (completely metrizable, separable space) the levels of this hierarchy do not collapse, that is $\mathbf{\Sigma}^0_\alpha \neq \mathbf{\Delta}^0_\alpha$ for all α. These levels of the Borel hierarchy thus give a way to measure the exact complexity of a set. We say a set A is $\mathbf{\Sigma}^0_\alpha$-hard if $A \notin \mathbf{\Pi}^0_\alpha$, which says that A is at least a complicated as sets at the $\mathbf{\Sigma}^0_\alpha$ level. We likewise say A is $\mathbf{\Pi}^0_\alpha$-hard if $A \notin \mathbf{\Sigma}^0_\alpha$. We say the set A is $\mathbf{\Sigma}^0_\alpha$-complete if $A \in \mathbf{\Sigma}^0_\alpha$ and is $\mathbf{\Sigma}^0_\alpha$, and likewise define A being $\mathbf{\Pi}^0_\alpha$-complete. Being $\mathbf{\Sigma}^0_\alpha$-complete says that the set is at the exact complexity of $\mathbf{\Sigma}^0_\alpha$, that is, it is $\mathbf{\Sigma}^0_\alpha$ but not any simpler (not $\mathbf{\Delta}^0_\alpha$). Similarly for being $\mathbf{\Pi}^0_\alpha$-complete. Computing upper-bounds for the complexity of a set is typically much easier than proving the lower-bounds.

In Theorems 2.2 and 2.4 we show that the set $\boldsymbol{O}$ of oracles separating $\boldsymbol{P}$ and $\boldsymbol{NP}$ is $\mathbf{\Pi}^0_2$-complete. We similarly show that $\boldsymbol{O'}$, the set of oracles separating $\boldsymbol{NP}$ and $\boldsymbol{co\text{-}NP}$ is $\mathbf{\Pi}^0_2$-complete. In Theorems 2.7 and 2.9 we show that the set $\boldsymbol{Q}$ of oracles relative to which $\boldsymbol{NP} \not\subseteq \boldsymbol{BQP}$ is also $\mathbf{\Pi}^0_2$-complete. Finally, in Theorems 2.12 and 2.13 we show that the set $\boldsymbol{S}$ of oracles separating $\boldsymbol{PSPACE}$ and $\boldsymbol{PH}$ is $\mathbf{\Pi}^0_2$-complete.

For the rest of this section we fix some notation we will use for the rest of the paper. We let $\omega = \mathbb{N} = \{0, 1, 2, \dots\}$ denote the natural numbers. The Baire space ω^ω is the space of functions from ω to ω with the product of the discrete topologies on ω. $[\omega]^\omega$ is the set of increasing function from ω to ω. 2^ω denotes the Cantor space of functions from ω to $2 = \{0, 1\}$ with the product of the discrete topologies. Both ω^ω and 2^ω are Polish spaces, with 2^ω being compact. We identify 2^ω with $\mathcal{P}(\omega)$, the power set of ω by considering characteristic functions. We also identify $[\omega]^\omega$ with the set of all infinite subsets of ω via their enumerating functions.

We fix for the rest of the paper an alphabet which we take to be the standard two-symbol alphabet $2 = \{0, 1\}$ unless otherwise specified. By a *string* we mean a finite sequence from the alphabet, that is, an element of $2^{<\omega}$. By a *language* we mean a subset of strings. We fix a reasonable effective enumeration of the strings, which we write as $s_0, s_1, \dots$. This way, the set of strings is identified with ω, and a language is identified with an element of 2^ω. An *oracle* refers to a particular language, that is, a specific subset of strings. Thus, oracles can also be identified with elements of 2^ω. As in the previous paragraph, we may also identify infinite languages or oracles with elements of $[\omega]^\omega$. In this way, natural notions such as measure or category on 2^ω apply to sets of oracles. Similarly the notion of Ramsey large on $[\omega]^\omega$ carries over to sets of infinite oracles. The exclusion of finite language or oracles in the $[\omega]^\omega$ case has no implications of significance regarding our results. Monotonicity is

a fundamental property of any reasonable notion of measure. If a class of languages turns out to be "large," in any sense, then it should and will remain large when we restrict to infinite oracles. Thus, throughout the paper we make no distinction between complexity classes and their slightly thinned out counterparts in Ellentuck space.

We introduce the classes of oracles we will be primarily concerned with in this paper. Recall that for a complexity class such as $\boldsymbol{P}$, $\boldsymbol{NP}$, etc., the relativized class $\boldsymbol{P}^A$, $\boldsymbol{NP}^A$ are are sets of languages that are in the class defined using A as an oracle. We will use the following notation for the sets of oracles providing separations of various classes.

$$\boldsymbol{O} = \{A : \boldsymbol{P}^A \neq \boldsymbol{NP}^A\} \tag{1}$$

$$\boldsymbol{O'} = \{A : \boldsymbol{NP}^A \neq \boldsymbol{co\text{-}NP}^A\} \tag{2}$$

$$\boldsymbol{S} = \{A : \boldsymbol{PH}^A \neq \boldsymbol{PSPACE}^A\} \tag{3}$$

$$\boldsymbol{Q} = \{A : \boldsymbol{NP}^A \not\subseteq \boldsymbol{BQP}^A\} \tag{4}$$

Before continuing to these results, we fix some common notations and conventions for later reference. If M is a Turing machine presumed to halt on all inputs, we will write $L(M)$ to denote the language decided by that machine. For a string x, let $|x|$ denote it's length. If σ is a symbol over some alphabet, we will write σ^n to denote the string that is simply σ repeated n times. We will for the rest of the paper fix a computable enumeration over the set of all deterministic Turing machines $\{M_i\}$, and another for all nondeterministic Turing machines $\{N_i\}$. For both enumerations, we assume that every machine appears infinitely often (perhaps with padding). We will commonly write TM as an abbreviation for Turing machine, and NTM as an abbreviation for nondeterministic Turing machine.

2. Descriptive Complexity

It turns out that most sets of oracles separating two different complexity classes are easily shown to be Π^0_2 sets–the following argument, applied first to $\boldsymbol{O}$, could easily be applied to many others, as we will see. For the hardness results in this section we will need to make use of a language powerful enough to force equality between the classes we are considering, when thought of as an oracle. Thus for the record we state the following lemma, whose proof can be found in any complexity theory textbook.

Lemma 2.1. *Let E be any $\boldsymbol{EXP}$-complete language. Then*

$$\boldsymbol{P}^E = \boldsymbol{NP}^E = \boldsymbol{BQP}^E = \boldsymbol{PH}^E = \boldsymbol{PSPACE}^E = \boldsymbol{EXP}$$

In the following subsections we determine the exact complexity of the sets of oracles separating the various pairs of classes we consider.

2.1. **P and NP.** First, we have the following upper-bound on the complexity of $\boldsymbol{O}$. A straightforward computation appears at first glance to give $\boldsymbol{O} \in \Sigma_3^0$. The following result shows it is actually Π_2^0.

Theorem 2.2. $\boldsymbol{O} \in \Pi_2^0$.

Proof. Define the following functions

$$\mathcal{N}_a(i,n,k) = \begin{cases} 1 & \text{if } N_i(x_n) \text{ halts in acceptance in time less than } |x_n|^k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\mathcal{N}_t(i,n,k) = \begin{cases} 1 & \text{if } N_i(x_n) \text{ halts in time less than } |x_n|^k \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$\mathcal{M}_a(j,n,k) = \begin{cases} 1 & \text{if } M_j(x_n) \text{ halts in acceptance in time less than } |x_n|^k \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\mathcal{M}_t(j,n,k) = \begin{cases} 1 & \text{if } M_j(x_n) \text{ halts in less than } |x_n|^k \text{ steps} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$\mathcal{M}_a$ and $\mathcal{M}_t$ are both clearly recursive (computable) To be more formal about $\mathcal{N}_t$ and $\mathcal{N}_a$: what we mean by "$N_i(n)$ halts in acceptance" is that there exists an accepting path in the configuration graph of $N_i(n)$. What we mean by "$N_j(n)$ halts in time less than $|n|^k$ steps" is that all paths of uniquely occurring vertices in the configuration graph of $N_i(n)$ (i.e., all of those paths without loops) have length less than $|n|^k$. Since both functions only require checking a finite number of paths, both of these functions $\mathcal{N}_a$ and $\mathcal{N}_t$ are recursive. For all of these functions, denote $\mathcal{M}_t^A$ and so forth to be the same functions, but with machines relativized to A.

The naive way to describe an oracle being in $\boldsymbol{O}$ is the following:

(9)

$$A \in \boldsymbol{O} \iff \exists i,k_1 \forall j,n_1 [\mathcal{N}_t^A(i,n_1,k_1) \wedge ((\exists k_2 \forall n_2 \mathcal{M}_t^A(j,n_2,k_2)) \Rightarrow \exists n_3 (\mathcal{N}_a^A(i,n_3,k_1) \neq \mathcal{M}_a^A(j,n_3,k_2)))]$$

This sentence makes the claim that there exists some relativized nondeterministic Turing machine N_i^A which halts in time polynomially bounded (with degree k_1) by its input lengths for all inputs (n_1 being the input), meaning that it decides some language $L_1 \in \mathbf{NP}^A$, and which has the property that if M_j^A is any polynomial time bounded Turing machine (of degree k_2) (meaning it decides some language $L_2 \in \mathbf{P}^A$), then there must exist an input (n_3) on which N_i^A and M_j^A disagree (meaning that $L_1 \neq L_2$). Putting this in prenex normal form pulls out another existential quantifier, yielding a Σ_3^0 expression for $\boldsymbol{O}$.

We can do better than this, however. The trick is to talk about equality of classes in terms of the existence of a "universal" language which, were it in the weaker of the two classes, would render that class just as powerful. We define such a language in terms of a "universal" nondeterministic Turing machine U. The machine U takes inputs of the form x, p^n, m, where x is a string, p is a special symbol which should never occur in x or m, and m codes an integer index for a nondeterministic Turing machine. On an input of this form, the machine U simulates the m^{th} nondeterministic machine on the input x, for $|x|+n$ many steps,

and accepts if the m^{th} machine accepts in this time, rejecting otherwise. Since there exists a universal nondeterministic Turing machine which operates in linear simulation overhead, with minor modification we can easily create a machine U which works as described above and halts in polynomial time. Thus the language decided by U, $L(U)$, is clearly in $\boldsymbol{NP}^A$, as is it's relativized counterpart $L(U^A)$ in $\boldsymbol{NP}^A$, for any fixed oracle A.

Suppose that for some oracle A, $L(U^A)$ were in $\boldsymbol{P}^A$. Let M_U^A be the poly-time machine which decides it. It is simple enough to check that $\boldsymbol{P}^A = \boldsymbol{NP}^A$: Let N^A be an arbitrary relativized nondeterministic Turing machine which halts in time n^k for some $k \in \omega$, and let m be the integer index of this machine. Fix an input x. To decide if $x \in L(N^A)$, we can simply simulate the machine M_U^A on the input $x; p^{|x|^k - |x|}; m$, and accept or reject accordingly. Clearly this process can be accomplished in polynomial time, and thus we have shown that $L(N^A) \in \boldsymbol{P}^A$. Thus we have that $L(U^A) \in \boldsymbol{P}^A$ iff $\boldsymbol{P}^A = \boldsymbol{NP}^A$. Let $u \in \omega$ be the index of the nondeterministic machine U, and let $l \in \omega$ be the polynomial bound on that machine. Stated in terms of the functions we have defined earlier, we have

(10)

$$A \in \boldsymbol{O}^c \iff \exists i, k \forall n [(\mathcal{M}_i^A(i,n,k) = 1) \wedge (\mathcal{N}_a^A(u,n,l) = 1 \iff \mathcal{M}_a^A(i,n,k) = 1)]$$

Thus $\boldsymbol{O}^c \in \Sigma_2^0$, and so $\boldsymbol{O} \in \Pi_2^0$. □

Since the claim that "there exists a machine which decides this language" is generally always going to be a Σ_2^0 statement, it's clear that such an argument should work for all complexity classes in which it's known that a universal machine exists. In particular, an identical argument confirms that $\boldsymbol{O'} \in \Pi_2^0$. The case of $\boldsymbol{Q}$ will also show itself to be a nearly identical argument, we give the details in section 2.2. The case of $\boldsymbol{S}$ requires a bit more discussion, but nonetheless we will show it to be Π_2^0 as well in section 2.3. Since the lightface classes are contained in their boldfaced counterparts, we have the following weaker corollary which will be useful to our Ramsey results:

Corollary 2.3. *$\boldsymbol{O}$ and $\boldsymbol{O'}$ are Borel.*

We next show that $\boldsymbol{O}$ and $\boldsymbol{O'}$ are $\boldsymbol{\Pi}_2^0$-hard, and thus are $\boldsymbol{\Pi}_2^0$-complete.

Theorem 2.4. *$\boldsymbol{O}$ is $\boldsymbol{\Pi}_2^0$ hard.*

Proof. We use the set

$$H = \{b \in 2^\omega : b(n) = 1 \text{ for infinitely many } n\}, \tag{11}$$

which is well-known (and easy to see) to be $\boldsymbol{\Pi}_2^0$-complete (see [7]). To show that $\boldsymbol{O}$ is $\boldsymbol{\Pi}_2^0$ hard it suffices to define a *reduction* f of H to $\boldsymbol{O}$. That is, $f : 2^\omega \to 2^\omega$ is a continuous function such that $b \in H$ iff $f(b) \in \boldsymbol{O}$. Note that despite having the same domain and range, we will be viewing elements of the domain as infinite binary sequences, and elements of the codomain as sets of strings over a fixed alphabet.

Thus given a binary sequence b, we want to construct an oracle A such that $\boldsymbol{P}^A \neq \boldsymbol{NP}^A$ iff b is 1 infinitely often. Let E be any $\boldsymbol{EXP}$-complete language. The strategy is to perform a "delayed" version of Solovay's original diagonalization out of $\boldsymbol{P}$ [11], in which the n^{th} stage of that construction only happens when we encounter the n^{th} 1 of the sequence. Whenever the n^{th} bit is a 0, we'll add all

strings from E of length i to the oracle, so that finitely many 1's corresponds to an oracle which is "as powerful" as E up to a finite number of answers which can be hard coded. Therefore by lemma 2.1, we will be left with an oracle A such that $\boldsymbol{P}^A = \boldsymbol{NP}^A$. With an infinite number of 1's, we will likely still be left with a significant amount of strings of E, but we will have nonetheless fully diagonalized out of $\boldsymbol{P}^A$.

We now describe the construction explicitly. Fix a bit string b. For any n, let $d(n)$ denote the number of 1's that have been encountered in b by the n^{th} digit (inclusive). We construct our oracle A in stages, corresponding to the bits of b. Let A_{n-1} be what we have constructed of the oracle after stage $n-1$. We commit that we will never add strings of length $i < n$ at stage n.

First, if $b(n) = 0$, then we simply add all strings from E of length n, and move to the next stage, i.e., A_n will equal A_{n-1} unioned with all strings of length n from E. Second, if $b(n) = 1$, then we run $d(n)^{th}$ Turing machine relativized to A_{n-1} for $n^{\log(n)}$ steps on the input 1^n. Suppose during this computation that the machine queries a string x. If $|x| < n$, then the decision of whether $x \in A_{n-1}$ will have already been settled. Suppose $|x| \geq n$. If this is the case and $b(|x|) = 1$, then we make note of this by adding x to an ongoing list of exceptions X (these are strings which we are not allowed to add at a later stage so as to not alter the computation). In running this computation, we proceed as if $x \notin A$, which will be the case if we follow our commitments regarding the set X. The goal of the stages corresponding to $b(n) = 1$ is to make it so that no polynomial time deterministic Turing machine can decide the language

$$L_A = \{1^n : A \text{ contains a string } x \text{ of length } n\} \tag{12}$$

We seek to do this by adding a single string x of length n to A_{n-1} in the case of the machine rejecting, and in the case of the machine not rejecting we add nothing, so as to ensure that the machine makes an error on 1^n. If the computation $M_{d(n)}^{A_{n-1}}(1^n)$ does not halt in rejection in at most $n^{\log(n)}$ steps then we add no new strings at stage n, that is, we set $A_n = A_{n-1}$. Suppose $M_{d(n)}^{A_{n-1}}(1^n)$ does halt in rejection within this number of steps. In the worst case scenario, every machine $M_1^{A_0}, \ldots, M_{d(n)}^{A_{n-1}}$ will have queried a different string of length n at every step of it's computation. In this case the set of strings X of length n which we've committed to not adding to our oracle is $\sum_{j=1}^{n} j^{\log(j)}$, which is less than 2^n for all $n \in \omega$. Thus, there will be a string x of length n which is not in the set X. We let $A_n = A_{n-1} \cup \{x\}$, where $x \notin X$ is the lexicographically least string not in X such that $|x| = n$. This completes the construction of the A_n. We let $A = \bigcup_n A_n$. We let $f(b)$ be the oracle A just constructed. We have thus defined the map $f : 2^\omega \to 2^\omega$.

To see this works, first suppose $b(n) = 1$ for only finitely many n. Then $A = (E \setminus P) \cup Q$ for some finite sets of strings P and Q. Clearly this is still $\boldsymbol{EXP}$ complete, and so $\boldsymbol{P}^A = \boldsymbol{NP}^A$. On the other hand, suppose that $b(n) = 1$ for infinitely many n. Suppose by way of contradiction that there is a polynomial time relativized deterministic Turing machine M^A which decides L_A. Let m be an index for it in our enumeration and k be the degree of it's polynomial bound. Let $j \in \omega$ be large enough that for all $i \geq j$, $i^k < i^{\log(i)}$. By our assumptions there exists an $l \geq j$ such that $b(l) = 1$ and $M_m^A = M_{d(l)}^A$. Thus at stage l of the construction, we

observed the computation of M_m^A on the input 1^l for a number of steps sufficient for it to halt and make a decision. Suppose it halted in rejection. Then at stage l we added a string x of length l to $A = f(b)$. So $1^l \in L_A$, a contradiction since M_m is supposedly deciding L_A. Suppose it halts in acceptance. Then by construction A has no strings of length l in it. Thus $1^l \notin L_A$, and our machine M_m is again in error. We have shown that no polynomial time machine can decide L_A.

On the other hand, $L_A \in \boldsymbol{NP}^A$ regardless of A, because we can simply have a nondeterministic machine guess a string of the proper length and check with the oracle to confirm membership, in $O(1)$ steps. Thus we have established that $\boldsymbol{P}^A \neq \boldsymbol{NP}^A$ iff $b(n) = 1$ for infinitely many n. That is, $b \in H$ iff $f(b) \in \boldsymbol{O}$.

It remains to show that this function is continuous. To show this it suffices to show that for any $b \in 2^\omega$, there exists a function $g{:}\,\omega \to \omega$ such that, for any string x, membership of x in $f(b)$ depends on at most $g(|x|)$ bits of b. Fix b and suppose x is a string of length n. We can decide if $x \in f(b)$ by running through the first n stages of the construction of $f(b)$. If $b(n) = 0$, then $x \in f(b)$ iff $x \in E$, meaning that only the first n bits of b are required to determine membership. If $b(n) = 1$, then we would add x to $f(b)$ iff the machine $M_{d(n)}^{f(b)}$ halts in rejection on 1^n, and x is the smallest string of length n that has not been queried at any stage $i \leq n$ such that $b(i) = 1$. It will likely be the case that this computation will need to query strings of length greater than n. However, there are finitely many such strings which are queried. Let $g(n)$ be the maximum length of these queried strings for this and the previous computations. Then membership of x in A depends only on the first $g(n)$ digits of b. □

Corollary 2.5. $\boldsymbol{O}$ *is* $\boldsymbol{\Pi}_2^0$*-complete.*

2.2. **NP and BQP.** It is clear that the argument of Theorem 2.4 can be potentially recycled for any pair of classes for which there is a diagonal process of constructing oracles separating them. Another example of this is the separation of $\boldsymbol{NP}$ from $\boldsymbol{BQP}$, and so we turn our attention to quantum Turing machines. The following results assume the definitions and conventions of Bernstein and Vazirani's oracle quantum Turing machine model from their landmark paper hammering out the technical details of these machines [3], and uses their notation. For a comprehensive overview of quantum Turing machines, we refer the reader to that paper. In the interest of completeness, however, we supply a brief overview. Quantum Turing machines are a generalization of the deterministic Turing machines (with two-way infinite tape) in which the transition function takes symbol-state pairs not to particular actions, but rather to vectors in which an efficiently computable complex number is assigned to every possible action which a deterministic machine could take in that context (the set of possible complex numbers which the transition function assigns can in fact be assumed finite). Configurations are seen as orthonormal basis vectors in a countable dimensional vector space populated by finite linear combinations of these. The machine with it's modified transition function implicitly defines a transformation from these basis vectors to finite superpositions of them, called the *time evolution operator*, and this can be extended to a full linear transformation on the space called the *linear time evolution operator*. The operator defined by the machine takes a fixed initial configuration to a finite superposition of possible configurations (one for each action, weighted according to the complex number given by the transition function), and further steps are seen as repeated

application of the same operator. The squared magnitude of the complex number weighting a configuration of the superposition is seen as giving the probability that one will observe that configuration upon "measurement" of the tape. The actual set of quantum Turing machines requires restrictions to be consistent with actual physics. In particular, the QTMs which should be viewed as legitimate are those for which the linear time evolution operator is unitary. Bernstein and Vazirani derive in their paper a finite set of local conditions directly on the transition function which are necessary and sufficient for this to be the case. This means that we can recursively enumerate over the set of all QTMs just as easily we can over DTMs, NTMs, or PH expressions.

Halting configurations (QTMs are defined to have a single halting state) deserve special mention. The convention is to say that a QTM halts in T steps if it after T applications of the linear time evolution operator, the machine is left in a superposition such that all non-zero configurations are halting configurations, and furthermore no halting configuration appears in any superposition up to that many applications. Thus the question of how to interpret superpositions in which some configurations are halting, and others are not, is avoided. Input conventions, i.e., specifying the initial configuration for a desired input string x, remain identical to deterministic Turing machines. A QTM is said to operate in time $f(n)$ if it halts on all strings of length n in time less than or equal to $f(n)$. A QTM is said to accept the string x with probability p in T steps if it halts in T steps from the appropriate initial configuration, and additionally if one measures the starting cell of the tape at this point, they will see a 1 with probability p, and a 0 with probability $1-p$ (we will assume that the alphabet for all QTMs include at the very least 0, 1, and a distinct blank symbol). Rejection is defined identically. An entire language L is accepted by a QTM with probability p if the machine accepts every string $x \in L$ with probability at least p, and also rejects every string $x \notin L$ with probability at least p. Finally the class $\boldsymbol{BQP}$ is defined to be the class of languages L for which there exists a QTM M, and an integer $k \in \omega$ such that M operates in time $O(n^k)$ and accepts L with probability $\frac{2}{3}$. Multitrack QTMs are also defined in the obvious way.

Turning to oracles, an oracle QTM has a special query track, and two special query states: a prequery state and postquery state. A query configuration is one in which the machine is in a prequery state, and the query tape has a single uninterrupted nonempty string (i.e., a single sequence of non-blank characters) of the form $x^\frown i$, where i is a 0 or a 1. Thinking of the oracle as a function outputting a 0 or a 1, application of the linear time evolution operator on a basis configuration of this form will always yield a configuration in which everything is unaltered except for the state, which becomes the postquery state, and the query tape, which will display x immediately followed by $i \bigoplus f(x)$, where $\bigoplus$ denotes addition modulo 2. This is important for keeping the transformation unitary, although for our purposes we will always assume that i is 0, so that the direct answer to the query is simply printed next the string we were querying. It should be noted that queries are deterministic–a basis state corresponding to a query configuration yields a single postquery basis configuration with amplitude 1. For an oracle A and a QTM M, M^A denotes the machine M relativized to A and $\boldsymbol{BQP}^A$ denotes the class of languages accepted with probability $\frac{2}{3}$ in polynomial time by a QTM relativized to A.

In accordance with the probabilistic convention for accepting and rejecting, simulation of one QTM by another, and simulation of arbitrary unitary operations by QTMs, are defined according to approximation rather than exactness. The following lemma from [1] is valuable to both our results and this idea.

Lemma 2.6 (Theorem 3.1 of [1])**.** *If two unit length superpositions $|\phi\rangle$ and $|\psi\rangle$ are within Euclidean distance ϵ of each other (i.e. $||\phi - \psi|| \leq \epsilon$), then the maximum probability distance between the probability distributions resulting from measurement is at most 4ϵ. That is to say, given any observable event E, the probability that E occurs upon measuring $|\phi\rangle$ will be no more than 4ϵ from the probability that E occurs upon measuring $|\psi\rangle$.*

By this lemma, two QTMs which are expected to have superpositions which are close in Euclidean distance, can be expected to be close in the distribution of measured outcomes. In their paper, Bernstein and Vazirani demonstrate that an arbitrary unitary transformations on n-qubit space can be approximated with exponential accuracy in polynomial time (weighted appropriately by the dimension n).

They also demonstrate the existence of a universal QTM, which can simulate arbitrary QTMs with any desired accuracy with slowdown polynomial in the runtime of the original machine and also polynomial in the inverse of the desired accuracy. Thus we can speak of simulating one QTM by another, or simulating an arbitrary unitary transformation. From these arguments it is also follows that there is a universal oracle Turing machine M^A that can efficiently simulate all other QTMs relativized to A.

The ability to recursively enumerate over the collection of all QTMs, along with the existence of universal relativized QTMs for any oracle, is enough to import our argument from Equation (10) into our current context. Redefine the context of $\mathcal{M}_a^A(j,n,k)$ and $\mathcal{M}_t^A(j,n,k)$ so that M_j refers to the j^{th} oracle QTM as opposed to the j^{th} oracle DTM, then by these two results we can be sure that these new functions are still recursive (relative to a fixed oracle). Have U, and the integer u be defined as in Equation (10). It is clear that $\boldsymbol{NP}^A \subseteq \boldsymbol{BQP}^A$ iff $L(U^A) \in \boldsymbol{BQP}^A$. We therefore have

(13)
$$A \in \boldsymbol{Q}^c \iff \exists i,k\,\forall n[(\mathcal{M}_t^A(i,n,k)=1) \wedge (\mathcal{N}_a^A(u,n,l)=1 \iff \mathcal{M}_a^A(i,n,k)=1)]$$

Thus we have

Theorem 2.7. ***Q*** *is Π_2^0 (and therefore also Borel).*

Superpositions of configurations will be denoted using 'bra-ket' notation. For a superposition $|\phi\rangle$ and a string y, the *amplitude measure* $q_y(\phi)$ is defined to be the sum of the squared magnitudes of all amplitudes corresponding to query configurations intending to ask the oracle about y. For our hardness results, as well as our results pertaining to Ramsey measure, we borrow strategies from Bennett, Bernstein and Vazirani's follow-up paper [2]. In particular we makes use of one of the lemmas found there.

Lemma 2.8 (Theorem 3.3 of [1])**.** *Fix a quantum Turing machine M, an oracle A, and an input x. For any natural number i, let $|\phi_i\rangle$ denote the current superposition for $M^A(x)$ at step i. Let $\epsilon > 0$. Let $F \subseteq [0, T-1] \times \{0,1\}^*$ be a set of time-string*

pairs such that $\sum_{(i,y)\in F} q_y(\phi_i) \leq \frac{\epsilon^2}{2T}$. *Suppose that for each* $(i,y) \in F$ *the answer to the query* $y \in A$ *at time* i *is changed to an arbitrary value. Let* $|\phi_i'\rangle$ *be the superposition at time* i *with these modified responses. Then* $||\phi_T\rangle - |\phi_T'\rangle| \leq \epsilon$.

See [1] for proofs of these results. This result, taken along with Lemma 2.6, imposes a bound on the extent to which a single string added to an oracle can change the overall superposition after a certain number of steps. We will use this to guarantee that by adding certain strings to our oracle, it won't "perturb" the superposition enough to flip any answers from acceptance to rejection or vice versa.

Theorem 2.9. $\boldsymbol{Q}$ *is* $\mathbf{\Pi}_2^0$ *hard.*

Proof. Fix an enumeration of all quantum Turing machines. Again fix an $\boldsymbol{EXP}$-complete problem E. Just as before, given a bit sequence $b \in 2^\omega$, we will conduct a step of diagonalization for each 1 in the sequence, and simply add appropriate strings from E otherwise. However, different from before is the number of strings we will potentially have to commit to not adding in order to maintain consistency of the construction. Normal deterministic Turing machines can only query polynomially many strings in a polynomial amount of runtime. However, when considering the superposition obtained from a polynomial number of steps of a quantum machine, we will typically be left with a superposition of exponentially many configurations, all of which could be intending to querying something different. We will encounter a similar problem for nondeterministic simulations. We can get around this difficulty by simulating our machines on very large inputs in comparison to the inputs simulated in previous stages of the construction. This will ensure that any string which we might want to add at a current stage will simply be too big to have appeared as a query in any simulation prior, leaving us free to add whatever we want. As before, we will be simulating for quasipolynomial runtimes. We fix an increasing function $g{:}\omega \to \omega$ such that for all n, $g(n+1) > g(n)^{\log(n)}$. This can be realized explicitly by $g(n) = 2^{2^{2^n}}$ (but 2^{2^n} is not enough!). This function g is fixed for the rest of the argument.

As before, we define a reduction f of H to $\boldsymbol{Q}$. Fix $b \in 2^\omega$, and we construct an oracle A which will be the value for $f(b)$. The oracle A is again constructed in stages, and A_n will denote the finite oracle constructed after stage n. We will have that the A_n are increasing with n. At stage n we will only add strings of lengths k with $g(n) \leq k < g(n+1)$.

If $b(n) = 0$, then we add to our oracle all strings from E of length k such that $g(n) \leq k < g(n+1)$. Note that if M is any QTM, then the superposition of $M(1^{g(n-1)})$ after at most $g(n-1)^{\log(g(n-1))}$ computation steps can only contain query configurations on strings bounded below this length, and this length by definition of g is less than the length of any strings we are adding at stage n. From this it will be clear that adding these strings from E will have no bearing on any prior simulation.

If $b(n) = 1$, then we run the $d(n)^{th}$ QTM on the input $1^{g(n)}$ for $T = g(n)^{\log(g(n))}$ many steps relativized to A_{n-1}, where $d(n)$ as before is the number of $m \leq n$ such that $b(m) = 1$. In constructing A_n we will add at most one string to A_{n-1}, and this string (if it exists) will be of length $g(n)$. Let $|\phi\rangle_i$ denote the superposition after i many steps, relativized to A_{n-1}. We look for the least $i \leq T$ such that $|\phi\rangle_i$ has a $> \frac{1}{2}$ probability of being in a halting state. If no such i exists, then we set $A_n = A_{n-1}$. Suppose such a time exists, and let i be the least such. If $|\phi\rangle_i$

has a $> \frac{1}{2}$ probability of being in a halting state which is accepting, then we also set $A_n = A_{n-1}$. If $|\phi\rangle_i$ has a $\leq \frac{1}{2}$ probability of being in a halting state which is accepting, then we set $A_n = A_{n-1} \cup \{y\}$, where y is the string constructed below such that $||\phi\rangle_i - |\phi'\rangle_i| < \frac{1}{28}$, where $|\phi'\rangle_i$ denotes the state after running the $d(n)^{th}$ QTM on the input $1^{g(n)}$ for i many steps relativized to $A_{n-1} \cup \{y\}$. Note that the string y will be too large to have appeared in any query at a previous stage of the construction. However, such a string could easily disturb the current superposition of interest $|\phi\rangle_i$.

We show the existence of a string y of length $g(n)$ as mentioned above. We will make use of lemma 2.8. With this goal in mind, Let S be the set of strings y of length $g(n)$ such that

$$\sum_{i=0}^{T-1} q_y(\phi_i) \geq \frac{1}{1568T} \tag{14}$$

Note first that it must be the case that $|S| \leq 1568T^2$. To see this, suppose that $|S| > 1568T^2$. Then we would have

$$T \geq \sum_{i=0}^{T-1} \sum_{y \in \{0,1\}^*} q_y(\phi_i) \geq \sum_{i=0}^{T-1} \sum_{y \in S} q_y(\phi_i) \tag{15}$$

$$\geq \sum_{y \in S} \frac{1}{1568T} \tag{16}$$

$$> 1568T^2 \frac{1}{1568T} = T \tag{17}$$

a contradiction. We wish to pull a string y of length $g(n)$ from the complement of this set S. In the worst case, all strings in S are of this length, but even then the total number of strings of length $2^{g(n)}$ far exceeds our upper bound for S, and so it follows that there exists a $y \notin S$. We take such string y (of length $g(n)$) in the definition of A_n above. This completes the definition of $A = f(b)$. It remains to show that f is a continuous reduction of H to $\boldsymbol{Q}$. The fact that f is continuous is straightforward as in the proof of Theorem 2.4. To finish the proof, we show that $b(n) = 1$ for infinitely many n iff A is in $\boldsymbol{Q}$. Note that regardless of b, the Solovay language L_A is in $\boldsymbol{NP}^A$.

If $b(n) = 1$ for has only finitely many n, then A equals E up to a finite difference, so that $\boldsymbol{BQP}^E = \boldsymbol{NP}^E$. Thus, $A = f(b) \notin \boldsymbol{Q}$.

Suppose next that $b(n) = 1$ for infinitely many n. Towards a contradiction, suppose there is a $\boldsymbol{BQP}^A$ machine M^A which decides L_A in time n^k for some k. Fix an i sufficiently large so that $g(i)^k < g(i)^{\log(g(i))}$. Since $b(n) = 1$ for infinitely many n, there is an $l \geq i$ such that $b(l) = 1$ and $M^A = M^A_{d(l)}$. Thus at stage l of the construction we simulated our machine $M^{A_{l-1}}$ on the input $1^{g(l)}$, for a number of steps large enough to ensure that M^A halts.

By assumption, M^A on $1^{g(l)}$ halts at some time $i_0 \leq T$. Let $|\phi'\rangle_i$ for $i \leq i_0$ denote the superposition from running M^A on $1^{g(l)}$ at time i. Note this is the same as running M^{A_l} on this input as strings of length $g(l+1)$ are too big to be queried. Let $|\phi\rangle_i$ for $i \leq i_0$ denote the state at time i from running $M^{A_{l-1}}$ on $1^{g(l)}$. By the conventions on quantum Turing machines, $|\phi'\rangle_{i_0}$ is a superposition

of configurations, each of which is in a halting state, and for all $i < \imath_0$, $|\phi'\rangle_i$ is a superposition of configurations, none of which is in a halting state. From the choice of y, Lemma 2.8 using $\epsilon = \frac{1}{28}$, and Lemma 2.6, that the probability that $|\phi\rangle_{i_0}$ is in a halting state is at least $1 - \frac{1}{7} > \frac{1}{2}$, but for $i < i_0$ the probability $|\phi\rangle_i$ is in a halting state is at most $\frac{1}{7} < \frac{1}{2}$. Thus, at step l of the construction, in defining A_l, we used time i. That is, $i = i_0$ is the least time so that $|\phi\rangle_i$ has a greater than $\frac{1}{2}$ probability of being in a halting state.

Suppose first that $M^A_{d(l)}(1^{g(l)})$ halts in an accepting state, that is, with at least a $\frac{2}{3}$ probability of acceptance. It follows from lemma 2.8 with $\epsilon = \frac{1}{28}$ and $F = \{0, 1, \ldots, T-1\} \times \{y\}$ that that $||\phi'\rangle_{i_0} - |\phi\rangle_{i_0}| \leq \frac{1}{28}$. By Lemma 2.6, the probability that $M^{A_{l-1}}_{d(l)}(1^{g(l)})$ accepts is within $\frac{4}{28} < \frac{1}{6}$ of that of $M^A_{d(l)}(1^{g(l)})$ accepting, and thus is greater than $\frac{1}{2}$. By the definition of A_l in this case we have that $A_l = A_{l-1}$, and so $y \notin A$. Thus the computation $M^A_{d(l)}(1^{g(l)})$ has made an error.

Suppose next that $M^A_{d(l)}(1^{g(l)})$ halts in a rejecting state, that is, with at least a $\frac{2}{3}$ probability of rejection. Then by the same argument of the previous paragraph, $M^{A_{l-1}}_{d(l)}(1^{g(l)})$ rejects with probability greater than $\frac{1}{2}$, and so $A_l = A_{l-1} \cup \{y\}$. So again $M^A_{d(l)}(1^{g(l)})$ has made an error.

This completes the proof that f is a continuous reduction of H to $\boldsymbol{Q}$. □

Corollary 2.10. $\boldsymbol{Q}$ *is* $\mathbf{\Pi}^0_2$ *complete.*

2.3. **PH and PSPACE.** Another famous diagonalization argument is the separation of $\boldsymbol{PH}$ and $\boldsymbol{PSPACE}$. In 1984, Furst, Saxe and Sipster [5] demonstrated that such a separation implied the existence of quasipolynomial size constant-depth circuits deciding the parity function. They themselves were able to demonstrate the non-existence of constant-depth polynomial size parity circuits, but were not able to rule out quasipolynomial ones. This was accomplished one year later by Yao [12], and then followed up by Håstad who proved the optimal lower bound via his famous switching lemma[6]:

Theorem 2.11 (Håstad)**.** *There is an increasing sequence* $\{n_k\}_{k\in\omega}$ *such that for all* k*, there are no depth* k *parity circuits on* $n > n_k$ *many inputs which are of size less than* $2^{\frac{1}{10}\frac{k}{k-1}n^{\frac{1}{k-1}}}$*.*

We will use this result to obtain our result that $\boldsymbol{S}$ is $\mathbf{\Pi}^0_2$-hard. We first fix some terminology and prove the upper-bound for the complexity of $\boldsymbol{S}$.

Let us call a *PH-expression* an expression of the form

$$\phi(y) = \exists^p x_1 \forall^p x_2 \ldots Q^p x_l R(x_1, x_2, \ldots, x_l, y) \tag{18}$$

where R is a polynomial time computable relation with degree k for some k, and the superscript p indicates that we are quantifying over strings of length less than or equal to $|y|^k$. An effective enumeration over all Turing machines can clearly be turned into an effective enumeration over all PH-expressions, for all l. A language is in $\boldsymbol{PH}$ iff it has a PH-expression defining membership. Likewise, by relativizing R to an oracle A, we obtain a definition of the relativized polynomial hierarchy $\boldsymbol{PH}^A$.

Theorem 2.12. $\boldsymbol{S} \in \Pi^0_2$ *(and therefore also Borel).*

Proof. It is well known that the quantified Boolean satisfiability problem, which we will denote $QSAT$, is $\boldsymbol{PSPACE}$ complete. However, with a small modification of the standard reduction witnessing that $QSAT$ is $\boldsymbol{PSPACE}$ hard, we can define a reduction from $\boldsymbol{PSPACE}^A$ to the "relativized language" $QSAT^A$, creating a version of the $QSAT$ problem which is always complete for $\boldsymbol{PSPACE}^A$ for all A. With this language defined, it is clear that $A \notin \boldsymbol{S}$ iff $QSAT^A \in \boldsymbol{PH}^A$, a statement that is easily expressible with a Σ_2^0 formula. To briefly recall how the normal reduction works: we have a Turing machine M operating in space n^k for some k, and we wish to create a family of quantified Boolean formulas

$$\phi(x) = \exists_{y_1} \forall_{y_2} \dots Q_{y_{|x|^k}} \psi(y_1, \dots, y_{n^k}, x)$$

such that x is in the language decided by M iff $\phi(x)$. Since our machine uses space n^k, we can be sure that the configurations of the machine are representable by strings of length polynomial in $|x|$. First, one define a base case $\psi_0(A, B)$ representing the claim that "configuration A yields configuration B in one step or that $A = B$." Then they define $\psi_i(A, B)$ to be the statement that there exists an intermediary configuration Z such that $\psi_{i-1}(A, Z)$ and $\psi_{i-1}(Z, B)$, i.e. that there exists a sequence of configurations from A to B of length less than or equal to 2^i. This is stated in a clever way such that only one "instance" of the ψ_{i-1} is needed, so that the number of configurations to be quantified over is kept linear in i. The only modification of this which is necessary is to add to the base case the extra complication necessary for checking that a string is in the oracle. That is to say, within the base case, where we normally have the Boolean formula representing the claim that "configuration A yields configuration B in zero or one steps," we have the longer claim:

- "configuration A is not in a query state and yields configuration B in zero or one steps, *or*
- A is in a query state, the string being queried is in A, and B is the configuration yielded by A given that the query accepts, *or*
- A is in a query state, the string being queried is not in A, and B is the configuration yielded by A given that the query rejects."

Clearly this requires no extra quantification and is only linearly longer than the original, and just as clearly the resulting expression will be $\boldsymbol{PSPACE}^A$ complete for any oracle A.

We have then that $A \notin S$ iff $QSAT^A \in \boldsymbol{PH}^A$. Let $\psi^A(x, y)$ be the formula representing that "relative to A, the x^{th} $\boldsymbol{PH}^A$ expression agrees with $QSAT^A$ on the input y," which can easily be seen as recursive using the appropriate modifications of our $\mathcal{M}$ functions from earlier. We have then that $A \notin \boldsymbol{S}$ iff $\exists x \forall y \psi^A(x, y)$, so that $\boldsymbol{S}^c$ is $\boldsymbol{\Sigma}_2^0$ complete. □

We now turn to the lower-bound for the complexity of $\boldsymbol{S}$.

By fixing a y, one can view a PH expression as in (18) as a constant depth circuit whose inputs correspond to strings which we may or may not decide to put in the oracle. For a fixed string length n and fixing membership in A for all other relevant strings of other length, we will see that these circuits are quasipolynomial size in it's number of inputs 2^n.

The way this works is as follows. At the top of the circuit is an OR gate, representing the leading $\exists$ quantifier. This gate has fanin $O(2^{n^k})$, where k is the polynomial bound of the PH-expression; one wire for every possible string of length

$\leq n^k$. Each of these wires is the lone output of an AND gate, representing the first $\forall$, which itself has fanin $O(2^{n^k})$ for the same reason. We continue like this until reaching a depth equal to the number of alternating quantifiers l. Note at this point that with y fixed, each bottom level gate corresponds to a particular selection of $x_1, \ldots, x_n$; all actual inputs to the PH-expression are fixed by this point. As mentioned, what we see as varying is not the inputs to the expression but rather the strings which may or may not be in the oracle A, and in particular those which are exactly length n (we will assume that all strings of relevant length not equal to n have been fixed as in or out of A, and therefore these wires can be seen as literals). In this way, the value of $R^A(x_1, \ldots, x_l, y)$ for any selection of inputs can be seen a function of some subset of the 2^n strings of length n which may or may not be in the oracle. Each bottom level gate representing the final quantifier can therefore be seen as being fed input from a DNF circuit in which each bottom level AND gate represents a particular selection of strings to explicitly have or not have in A so as to ensure a positive result. Finally we note that the fanin of these parent OR gates can all be assumed the same as more or less the same as the others: $O(2^{n^k})$. The reason for this is that when a machine makes a query, it's computation goes in one of exactly two "directions," and at most polynomially many such choices are made. Thus one can imagine a binary tree of polynomial depth, where some number of these results in acceptance. Since there are at most $O(2^{n^k})$ many accepting "paths," there are also at most 2^{n^k} many combinations of strings which need to be explicitly specified to ensure acceptance of the machine. We are therefore left with a depth $l+1$ circuit which takes $m = 2^n$ many inputs, with size $O(2^{l \log^k(m)})$, which is equivalent to the polynomial expression. The quasipolynomial size is sufficient to be sure that the circuits in question cannot decide parity, and this ensures the existence of a set of strings which can be added to the oracle such that the parity of it doesn't match the circuit's output.

Specifically, for a given oracle A we consider the language

$$L_A = \{1^j : \text{ there is an even number of strings of length } j \text{ in } A\}.$$

Note that L_A is in $\boldsymbol{PSPACE}^A$ for any A, since one can simply keep a tally using linear space while repeatedly querying every length n string. We will use the above circuit construction and Theorem 2.11 diagonalize out of $\boldsymbol{PH}^A$ while staying inside of $\boldsymbol{PSPACE}^A$ when $b \in 2^\omega$ is in H. We use an argument similar to that of the previous theorems.

Theorem 2.13. *$\boldsymbol{S}$ is $\boldsymbol{\Pi}^0_2$ hard.*

Proof. As before, fix an $\boldsymbol{EXP}$-complete language E. We will define a continuous mapping $f: 2^\omega \to 2^\omega$ such that a string b is in H iff $\boldsymbol{PH}^{f(b)} \neq \boldsymbol{PSPACE}^{f(b)}$, where H is the collection of strings with infinitely many 1's just as before. The proof will be similar in strategy to what we've done twice before: perform a delayed diagonalization in which we only do the next stage upon encountering a 1 for the fixed bit string b which we are mapping to an oracle A. Let $g(n)$ be an increasing function which grows fast enough to ensure that the circuit representing a PH-expression on $1^{g(n)}$ can never decide parity by 2.11, and that no strings of length at least $g(n)$ could possibly appear as relevant to circuits constructed in previous stages. For the first condition it will suffice to have $2^{\log(n) g(n)^{\log(n)}} < 2^{\frac{1}{100} 2^{g(n)/\log(n)}}$,

which will hold for any increasing g for all large enough n. For the second condition it suffices to have $g(n+1) > g(n)^{\log(n)}$, as in Theorem 2.9. So, $g(n) = 2^{2^{2^n}}$ suffices.

We describe a stage n of the construction. First of all, regardless of the value of $b(n)$, we will always add to A all strings from E of lengths k with $g(n) \leq k < g(n+1)$. When $b(n) = 0$, this is all we do. Now suppose $b(n) = 1$. In this case, we consider the $d(n)^{th}$ PH expression ϕ as in Equation 18, where again $d(n)$ is the number of $m \leq n$ with $b(m) = 1$. Say that $k \in \omega$ is the polynomial bound of the expression, and l the number of quantifier alternations. Create as described above a circuit representing the PH-expression, with the input y fixed as $1^{g(n)}$, which has depth $l+1$ and size $O(2^{l \log^k(m)})$ where $m := 2^{g(n)}$, and whose inputs correspond to strings of length $g(m)$ which we may or may not decide to add to the oracle. We are promising as before to not add any strings of length not equal to $g(n)$ at stage n of the construction. By this commitment, along with a commitment to not add any strings of lengths between $g(n)$ and $g(n+1)-1$ besides those from E, we can assume that strings of length $g(n)$ are the only non-literal inputs to the circuit. By construction we can be sure that this circuit fails to decide the parity function for strings of length $g(n)$. We use here the fat that the PH expression ϕ will be considered at infiitely many stages n, and that for large enough such n we will have that $2^{l \log^k(m)} < 2^{\log(n) \log^{\log(n)}(m)} = 2^{\log(n) g(n)^{\log(n)}}$ which is less than $2^{\frac{1}{100} 2^{g(n)/\log(n)}}$, which is below the bound of Theorem 2.11. This means that there must exist an assignment to the inputs such that the output of the circuit cannot be the parity of the assignment, i.e., there exists a collection of strings of length $g(n)$ which we can add to A_{n-1} so that the $d(n)^{th}$ PH-expression cannot be deciding the test language

$$L_A = \{1^n : \ A \text{ has an even number of strings of length } n\}$$

This completes the description of A. Define the function f by setting $f(b) = A$, where $A = \bigcup_n A_n$ is the oracle constructed from b. It is clear that if b has an infinite number of 1's, then we will ensure that every PH-expression disagrees with the language $L_{f(b)}$ on at least one input, and therefore $L_A \notin \boldsymbol{PH}^A$. On the other hand, if b has only finitely many 1's, then A will be equal to E up to a finite difference, and thus $\boldsymbol{PH}^{f(b)} = \boldsymbol{PSPACE}^{f(b)}$. Thus we have confirmed that $b \in H$ iff $\boldsymbol{PSPACE}^{f(b)} = \boldsymbol{PH}^{f(b)}$. As before, continuity is clear from the observation that deciding if a string is in $f(b)$ only requires a finite initial segment of b, which is clear from the construction. □

Corollary 2.14. *$\boldsymbol{S}$ is $\boldsymbol{\Pi}^0_2$-complete.*

2.4. **The Descriptive Complexity of More Common Classes.** Lastly, we consider the descriptive complexity of common complexity classes themselves. Since these collections of oracles are countable, they cannot be Π^0_2 complete as in our previous theorems. However, it is quite easy to demonstrate that virtually all of them are Σ^0_2 and $\boldsymbol{\Sigma}^0_2$-complete. We will consider the class $\boldsymbol{P}$ as our prototypical example. It is easy to see that $\boldsymbol{P}$ is naturally a Σ^0_2 set. Simply note that a language is in $\boldsymbol{P}$ iff there exists a Turing machine exists which agrees with the language on all inputs. So, $A \in P$ iff $\exists i \ \forall n \ [(n \in A) \leftrightarrow M_i(n) = 1)]$, which shows $\boldsymbol{P} \in \boldsymbol{\Sigma}^0_2$. The same simple computations works for the other complexity classes.

It is also fairly simple to show $\boldsymbol{\Sigma}^0_2$ hardness for $\boldsymbol{P}$ and the other classes. For example, we prove the following.

Theorem 2.15. *$\boldsymbol{P}$ is $\boldsymbol{\Sigma}^0_2$-complete.*

Proof. As the set H of bit strings which are 1 infinitely often is $\mathbf{\Pi}^0_2$-complete, the set $2^\omega \setminus H$ of bit strings which are 1 finitely often is $\mathbf{\Sigma}^0_2$-complete. We define a reduction f of $2^\omega \setminus H$ to $\boldsymbol{P}$. For any $b \in 2^\omega$ we define $f(b) = L$ as follows. Let $d(i)$ be the number of 1s which have occurred by the i^{th} bit. We create L in stages. At stage n, if $b(n) = 0$ we set $L_n = L_{n-1}$. If $b(n) = 1$, we run the $d(n)$th polynomial time machine on input 1^n for time $n^{\log(n)}$. If that computation terminates with a rejection, then we let $L_n = L_{n-1} \cup \{1^n\}$, and otherwise set $L_n = L_{n-1}$. If b has finitely many 1s, then we will only have added a finite number of strings to L so clearly $L \in \boldsymbol{P}$. On the other hand if $b(n) = 1$ for infinitely many n, then we have diagonalized against all polynomial time machines (recall that our enumeration of polynomial time machines repeats each machine infinitely often, and $\log(n)$ will eventually be larger than any fixed k). Thus the language L created is in $\boldsymbol{P}$ iff $b \in 2^\omega \setminus H$.

□

It is clear that the argument just supplied for $\boldsymbol{P}$ works for other complexity classes. We have the following, for example.

Theorem 2.16. ***NP***, ***EXP***, ***co-NP***, ***BQP***, ***PH***, ***L***, ***NL***, ***PSPACE*** *are all* $\mathbf{\Sigma}^0_2$*-complete.*

3. Ramsey Theory

The notion of *Ramsey large* is, aside from measure and category, another important measure of size for sets $S \subseteq [\omega]^\omega$. Recall we are identifying infinite subsets of ω with increasing functions $f : \omega \to \omega$. We let S^c denote the complement $[\omega]^\omega \setminus S$. One way to define this notion is through the *Ellentuck topology.* Letr a be a finite subset of ω and $A \subseteq \omega \setminus s$ an infinite set. Then a basic open set in the Ellentuck topology is a set of the form

$$[a, A] = \{S \in [\omega]^\omega : S \setminus A = a \wedge S \setminus a \subseteq A\}$$

If we choose to view the infinite subsets of ω as increasing functions, then

$$[a, A] = \{f \in [\omega]^\omega : \forall i < |a|(a(i) = f(i)) \wedge \forall i \geq |a|(f(i) \in A)\}$$

The Ellentuck topology is closely related to *Mathias forcing* in set theory, where the objects $[a, A]$ as above are the elements of the forcing partial order. The Ellentuck topology/ Mathias forcing arises naturally in the study of partition properties on ω. Recall the classical Ramsey theorem says that for any partition $P : \omega^m \to k$, where $m, k \in \omega$, there is an infinite set $H \subseteq \omega$ which is homogeneous for the partition. That is, $P \restriction [H]^\omega$ is constant. With the axiom of choice, no infinite cardinal, including ω, can have an infinite exponent partition property, but if we restrict to reasonable definable sets $S \subseteq [\omega]^\omega$, then such partition relations can hold. The Galvin-Prikry theorem, stated next, says that Borel partition of $[\omega^\omega]$ have homogeneous sets.

Theorem 3.1 (Galvin-Prikry)**.** *Let* $[\omega]^\omega = P_0 \cup P_1 \cup \ldots \cup P_{k-1}$ *be a partition of* $[\omega]^\omega$*, with each* P_i *is Borel (in the usual topology on the Baire space). Then there exists an infinite* $H \subseteq \omega$ *which is homogeneous for the partition. That is, there is an* $i < k$ *such that* $[H]^\omega \subseteq P_i$*.*

Silver extended this result to include all $\mathbf{\Sigma}^1_1$ partitions. Assuming Woodin's axiom AD^+, every partition P of $[\omega]^\omega$ into finitely many pieces has a homogeneuous set. For this paper, however, it will be enough to consider Borel partitions.

A set $S \subseteq [\omega]^\omega$ is called *Ramsey* if there is an $H \in [\omega]^\omega$ such that $[H]^\omega \subseteq B$, or $[H]^\omega \subseteq [\omega]^\omega \setminus B$. In other words, S, viewed as a partition of $[\omega]^\omega$ into two pieces, has a homogeneous set. Thus, the Galvin-Prikry theorem asserts every Borel set in $[\omega]^\omega$ is Ramsey. A somewhat stronger notion is that of S being *completely Ramsey*. This means that for every basic open set $[a, A]$ we have an infinite $H \subseteq A$ such that $[a, H] \subseteq S$ or $[a, H] \subseteq S^c$. In fact, the Galvin-Prikry theorem asserts that every Borel set $S \subseteq [\omega]^\omega$ is completely Ramsey.

A basic fact about the Ellentuck topology is that a set $S \subseteq [\omega]^\omega$ is nowhere dense iff for every basic open set $[a, A]$ there is a $B \in [A]^\omega$ such that $[a, B] \subseteq S^c$. This differs from the definition of nowhere dense in that we do not need to change the stem a, but just need to thin out the set A. Also a set $S \subset [\omega]^\omega$ is meager iff it is nowhere dense. Thus, a set S is meager iff for every $[a, A]$ there is a $B \in [A]^\omega$ such that $[a, B] \subseteq S^c$. The meager sets, as usual, form a σ-ideal on $[\omega]^\omega$.

Recall that a set S in a topological space has the Baire property iff there is an open set U such that $S \triangle U$ is meager. We have the following characterization of the completely Ramsey sets.

Theorem 3.2 (Ellentuck)**.** *A set $B \subseteq [\omega]^\omega$ is completely Ramsey iff it has the Baire property in the Ellentuck topology.*

We say a set $S \subseteq [\omega]^\omega$ is Ramsey small (or Ramsey measure 0) iff it is meager in the Ellentuck topology, and call it Ramsey large (or Ramsey measure 1) if it is comeager in this topology. A set which is not Ramsey small is called Ramsey positive. Note that a set S with the Baire property in the Ellentuck topology is Ramsey positive iff it contains a non-empty basic open set $[a, A]$.

In this section we appply the notion of Ramsey largeness to the study of oracles separating complexity classes. We have two goals in mind in proving these results. The first is to generalize the known results concerning the largeness of these collections with respect to measure and category. The second is to provide a possible plausible version of the random oracle hypothesis, which we discuss below.

We first show that the set $\boldsymbol{O'}$ of oracles (viewed as infinite subsets of ω) K such that $\boldsymbol{NP}^K \neq \boldsymbol{co\text{-}NP}^K$ is Ramsey positive. Recall we are identifying oracles K with subsets of ω by our fixed enumeration of the finite strings.

Theorem 3.3. *The set $\boldsymbol{O'}$ is Ramsey positive. That is, there exists a homogeneous $H \subseteq \omega$ such that for all infinite oracles K consisting of strings from H, $\boldsymbol{NP}^K \neq \boldsymbol{co\text{-}NP}^K$.*

Proof. We established in §2 that $\boldsymbol{O'}$ is a Borel set. By the Galvin-Prikrey theorem, this means that the set is completely Ramsey. To show $\boldsymbol{O'}$ is Ramsey positive it suffices to get an $A \in [\omega]^\omega$ such that $[A]^\omega \subseteq \boldsymbol{O'}$. By Galvin-Prikry it suffices to construct an A such that for every infinite $B \subseteq A$ there is an infinite $C \subseteq B$ such that $\boldsymbol{NP}^C \neq \boldsymbol{co\text{-}NP}^C$. We proceed to construct such a set A.

We fix a sufficiently fast growing function $f{:}\,\omega \to \omega$, say with $f(n+1) > f(n)^{\log(f(n))}$, and we may also assume that $n2^{n-1}f(n)^{\log(f(n))} < 2^{f(n)}$as the function 2^x grows faster than $x^{\log(x)}$. Our set A will have exactly 1 string of length $f(n)$ for all n. We construct A in stages, and refer to the partially constructed set at stage n by A_n. At stage n, we run for each $m \leq n$ and each subset $S \subseteq A_{n-1}$, the

mth $\boldsymbol{NP}$ machine on the partial oracle S with input $1^{f(n)}$ for $f(n)^{\log(f(n))}$ many steps. For each such pair (m, S), if this run of the mth $\boldsymbol{NP}$ machine results in an acceptance, then there is a valid path in the configuration graph for the machine of length at most $f(n)^{\log(f(n))}$ which terminates in an acceptance state. This particular computation path queries at most $f(n)^{\log(f(n))}$ many strings of length $f(n)$. We fix a set $E(m, s)$ of strings of length $f(n)$ and $|E(m, S)| \leq f(n)^{\log(f(n))}$ which contains this set of queries. Note that in this case that if A' agrees with A_{n-1} on all strings of length $< f(n+1)$ except for strings of length $f(n)$, and if $A' \cap E(m, S) = \emptyset$, then the mth $\boldsymbol{NP}$ machine run with the oracle A' (with input $1^{f(n)}$ for $f(n)^{\log(f(n))}$ many steps) will also terminate in acceptance. Let $E_n = \cup\{E(m, S) : m \leq n \wedge S \subseteq A_{n-1}\}$. We have $|E_n| \leq n2^{n-1} f(n)^{\log(f(n))} < 2^{f(n)}$. Thus there is a string s_n of length $f(n)$ with $s_n \notin E_n$. We let $A_n = A_{n-1} \cup \{s_n\}$. We let $A = \bigcup_n A_n$. This completes the definition of the set A.

Suppose now $B \in [A]^\omega$ is an infinite subset of A. We show that there is a $C \in [B]^\omega$ such that $\boldsymbol{NP}^C \neq \boldsymbol{co\text{-}NP}^C$. Let $g(n)$ be the nth element of B. Clearly $g(n) \geq f(n)$. We construct $C \subseteq B$ by a diagonalization argument similar to our previous proofs. We construct C in stages and let C_i be the set constructed at stage i. We will have $C_i \subseteq B \cap [0, g(i)]$. Assume C_{n-1} has been defined. We run the nth $\boldsymbol{NP}$ machine on the input $1^{g(n)}$ for $g(n)^{\log(g(n))}$ steps relative to the oracle C_{n-1}. If this computations halts in an acceptance, that is there is an accepting path in the configuration graph of the non-deterministic computations, then we let $C_n = C_{n-1} \cup \{s_n\}$, where s_n is the unique string of length $g(n)$ in B. Otherwise we set $C_n = C_{n-1}$.

It remains to show that $\boldsymbol{NP}^C \neq \boldsymbol{co\text{-}NP}^C$. The language witnessing this difference will be the standard Solovay language L_C (the set of strings 1^k such that there is a string of length k in C), which as always is clearly in $\boldsymbol{NP}^C$. Suppose it were also in $\boldsymbol{co\text{-}NP}^C$. Then *co*-$L_C$ (the complement of L_C) would be in $\boldsymbol{NP}^C$. That is, there would exist polynomial time (say time n^k for a fixed $k \in \omega$) relativized nondeterministic machine N^C which accepts it. Pick an m sufficiently large so that $g(m)^{\log(g(m))} > g(m)^k$, and where the mth non-deterministic machine N_m^C is N. Consider the operation of this machine, run for $g(m)^{\log(g(m))}$ steps, on input $1^{g(m)}$ relative to the oracle C_{n-1}. Suppose first this computations halts with an acceptance. Then $C_n = C_{n-1} \cup \{s_n\}$, and s_n is not queried in some accepting path for the nondeterministic computation. Thus $N_m^{C_n}(1^{g(n)})$ also halts with an acceptance. Since $g(n+1) \geq f(n+1) \geq f(n)^{\log(f(n))}$, it then follows that $N_m^C(1^{g(n)})$ also halts with an acceptance. Since there is a string of length $g(n)$ in C, namely s_n, the machine N_m^C has made an error on the input $1^{g(m)}$. Suppose next that $N_m^{C_{n-1}}(1^{g(n)})$ does not halt with an acceptance when run for $g(m)^{\log(g(m))}$ many steps. So, $C_n = C_{n-1}$. So, $N_m^{C_n}(1^{g(m)})$ also does not halt with an acceptance, and likewise $N_m^C(1^{g(m)})$ does not halt with an acceptance when run this many steps. Since $1^{g(m)} \notin L_c$ in this case, the machine N_m^C has made an error on the input $1^{g(m)}$ in this case as well. We have confirmed that no polynomial time relativized nondeterministic Turing machine can possibly decide *co*-L_C, completing the proof. □

Since $\boldsymbol{O'} \subseteq \boldsymbol{O}$, we immediately have the following.

Corollary 3.4. *Both $\boldsymbol{O}$ and $\boldsymbol{O'}$ are Ramsey positive.*

A similar argument applied on top of the argument from Theorem 2.9 will separate $\boldsymbol{NP}$ from $\boldsymbol{BQP}$ according to the following theorem.

Theorem 3.5. *The set $\boldsymbol{Q}$ is Ramsey positive. That is, there exists a homogeneous set $H \in [\omega]^\omega$ such that for any $C \in [H]^\omega$ we have $\boldsymbol{NP}^C \not\subseteq \boldsymbol{BQP}^C$.*

Proof. As in the proof of Theorem 3.3 we construct an infinite set $A \subseteq \omega$ (again identified with a set of strings) such that for any $B \in [A]^\omega$ there is a $C \in [B]^\omega$ such that $\boldsymbol{NP}^C \not\subseteq \boldsymbol{BQP}^C$. We fix a fast growing $f: \omega \to \omega$ satisfying $f(n+1) > f(n)^{\log(f(n))}$ and $n2^n 1568 f(n)^{2\log(f(n))} < 2^{f(n)}$ for all n. We again define A in stages, and let A_n be the set defined at stage n. We will get A_n from A_{n-1} by adding at most one string of length $f(n)$. Let M_m enumerate all of the quantum Turing machines, with each machine repeated infinitely often. At stage n, suppose A_{n-1} has been defined. For each $m \leq n$ and each $S \subseteq A_{n-1}$, consider running the mth quantum machine M_m on the input $1^{f(n)}$ relative to the oracle S for $T = f(n)^{\log(f(n))}$ many steps. Let $|\phi\rangle_t^{m,S}$ for $t \leq T$ denote the superposition at time t. If y is a string of length $f(n)$, let S_y denote the oracle $S_y = S \cup \{y\}$. Let $E(m,S)$ denote the set of strings y of length $f(n)$ such that $\sum_{i=0}^{T-1} q_y(|\phi\rangle_i^{m,S}) < \frac{1}{1568T}$, where as in Theorem 2.9 $q_y(|\phi\rangle_i^{m,S})$ the the sum of the squared magnitudes of the amplitudes of configurations in the superposition $|\phi\rangle_i^{m,S}$) corresponding to a query state in which the string y is being queried. In the proof of Theorem 2.9 we showed that $|E(m,S)| \leq 1568T^2$. So we have:

$$|\bigcup_{m,S} E(m,S)| \leq n2^n 1568T^2 = n2^n 1568 f(n)^{2\log(f(n))} < 2^{f(n)}.$$

Let $A_n = A_{n-1} \cup \{y_n\}$ where y_n is a string of length $f(n)$ not in $E_n = \bigcup_{m,S} E(m,S)$. This completes the definition of the set $A = \bigcup_n A_n$.

Suppose now $B \in [A]^\omega$. We construct a $C \in [B]^\omega$ such that $\boldsymbol{NP}^C \not\subseteq \boldsymbol{BQP}^C$. In fact, we will have that the Solovay language

$$L_C = \{1^k : \text{there is a string of length } k \text{ in } C\}$$

is not in $\boldsymbol{BQP}^C$.

Let $g(n)$ be the nth element of B, so $g(n) \geq f(n)$. We construct C in stages as before, and at stage n we get C_n from C_{n-1} by adding at most one string of length $g(n)$, which would have to be the unique string in B of length $g(n)$. Suppose we are at stage n and C_{n-1} has been defined. Let $S = B \cap [0, g(n-1)]$. Consider the quantum machine M_n run on input $1^{g(n)}$ relative to the oracle S for $T = g(n)^{\log(g(n))}$ many steps. Let $|\phi\rangle_t$ be the superposition of this computation at time $t \leq T$. Suppose first that there is a time $t \leq T$ such that $|\phi\rangle_t$ has a greater than $\frac{1}{2}$ probability of being in a configuration which is a halting state which is a rejecting state. In this case we ste $C_n = C_{n-1} \cup \{s_n\}$ where s_n is the unique string in B of length $g(n)$. In all other cases we set $C_n = C_{n-1}$. The completes the definition of C.

Suppose towards a contradiction that the Solovay language L_C were in $\boldsymbol{BQP}^C$. Let M be a quantum Turing machine, say with run time bounded by n^k for input size n, which decides the language L_C. Fix m such that $M_m = M$ and m is large enough so that $\log(m) > k$. Consider stage m in the construction of C. Let $|\phi'\rangle_t$ for $t \leq T$ be the superposition at time t for the quantum computation $M_m^C(1^{g(m)})$, and let $|\phi^{s_m}\rangle_t$ denote the superposition for the computation $M_m^{C_{n-1} \cup \{s_m\}}(1^{g(m)})$ where $s_n m \in A$ is the unique string in A of length $g(m)$. Since $g(m+1) \gg g(m)$ we have (since C_m is either C_{m-1} or $C_{m-1} \cup \{s_m\}$) that $|\phi'\rangle_t$ is either $|\phi\rangle_t$ or $|\phi^{s_m}\rangle_t$.

Suppose first that $M_m^C(1^{g(m)})$ accepts, that is, there is a time $t_0 \leq T$ such that $|\phi'\rangle_{t_0}$ is in a superposition of halting configurations and there is a $> \frac{2}{3}$ probabilty that these halting states are accepting. Furthermore, in this case we must have that for all $t' < t_0$ that $|\phi'\rangle_{t'}$ is in a superposition of configurations, none of which is in a halting state (from the definition of a well-formed quantum Turing machine halting, see [3]). For such t', $|\phi\rangle_{t'}$ has a $< \frac{1}{2}$ probability of being in a halting state since from the choice of s_m in the construction of A and Lemma 2.8 we have that $||\phi\rangle_{t'} - |\phi'\rangle_{t'}| < \frac{1}{28}$ and so from Lemma 2.6 we have that the difference between the probability of $|\phi\rangle_{t'}$ being in a halting state and $|\phi'\rangle_{t'}$ being in a halting state is at most $\frac{1}{7}$. Thus, the time t used in the definition of C_m is equal to t_0. Also, the probability that $|\phi\rangle_{t_0}$ is in a halting state which rejects the input is at most $\frac{1}{3} + \frac{1}{7} < \frac{1}{2}$. So by definition of C_m we have in this case that $C_m = C_{m-1}$, and so $1^{g(m)} \notin L_C$. Thus, the machine $M = M_m$ has made an error.

Suppose next that $M_m^C(1^{g(m)})$ rejects, and let t_0 again be the corresponding time. As in the previous paragraph, the time t as in the definition of C_m is equal to t_0. Also as in the previous paragraph, the probability that $|\phi\rangle_{t_0}$ is a halting configuration which rejects the input is at least $\frac{2}{3} - \frac{1}{7} > \frac{1}{2}$. So by definition of C_m we have $C_m = C_{m-1} \cup \{s_m\}$, so $1^{g(m)} \in L_A$. Thus again the machine M has made an error on the input $1^{g(m)}$.

In both cases we conclude that it cannot be the case that M actually decides L_C. Thus it cannot be the case that $L_C \in \boldsymbol{BQP}^C$, and so $\boldsymbol{NP}^C \not\subseteq \boldsymbol{BQP}^C$.

□

Surprisingly, the analogous statement for $\boldsymbol{S}$, that $\boldsymbol{S}$ is Ramsey positive, appears to be stronger than the other two. We discuss this further in §3.1 below.

3.1. **The Ramsey Oracle Hypothesis.** In this section we present results which appear to suggest that the Ramsey version of the random oracle hypothesis may hold. That is, if two complexity classes are equal when relativized to a Ramsey large set of oracles, then the two unrelativized classes are equal. Note in this regard that Theorems 3.3 and 3.5 only show that for the classes considered there that the set of separating oracles is Ramsey positive. We will show for the classes considered in those results that if the collection of separating oracles is actually Ramsey large, then the corresponding unrelativized classes are equal. For the set $\boldsymbol{S}$ of oracles separating $\boldsymbol{PSPACE}$ and $\boldsymbol{PH}$, we will show the stronger result that if $\boldsymbol{S}$ is just Ramsey positive, then $\boldsymbol{PSPACE} \neq \boldsymbol{PH}$.

The following result demonstrates that showing that $\boldsymbol{O}$ is Ramsey large is at least as difficult as the $\boldsymbol{P} = \boldsymbol{NP}$ problem itself.

Theorem 3.6. *If the set $\boldsymbol{O}$ of oracles separating $\boldsymbol{P}$ and $\boldsymbol{NP}$ is Ramsey large then $\boldsymbol{P} \neq \boldsymbol{NP}$.*

Proof. Assume $\boldsymbol{P} = \boldsymbol{NP}$ and we produce an infinite set $A \subseteq \omega$ (which we are identifying with a set of strings) such that for any $B \in [A]^\omega$ we have $\boldsymbol{P}^B = \boldsymbol{NP}^B$, which contradicts $\boldsymbol{O}$ being Ramsey large. Let $A = \{1^n : n \in \omega\}$, or more generally we can take $A = \{1^{f(n)} : n \in \omega\}$ where $f : \omega \to \omega$ is any increasing function f. Suppose $B \in [A]^\omega$. We show $\boldsymbol{NP}^B = \boldsymbol{P}^B$. Let N^B be a relativized nondeterministic Turing machine which runs in polynomial time, say bounded by n^k where n is the input length, and which decides a language L. We describe a deterministic machine M such that M^B runs in polynomial time and decides L. First, there is a deterministic

machine M_1^B which on an input of length n computes a "look-up table" for B, that is, it records for all strings s of the form 1^m, for $m \leq n^k$, whether or not $s \in B$. Of course, for strings s not of this form, we know that $s \notin B$. Second, there is an unrelativized nondeterministic machine N_2 which takes as input a string s of length n and a string t of length $\leq n^k$ and simulates the machine N in the following sense. Whenever a computation path of the nondeterministic machine N would query the oracle on a string u, the machine N_2 proceeds as if the answer is "no" if u is not of the form 1^m, and if u is of this form then N_2 uses the answer "yes" iff u has length $\leq n^k$ and $t(m) = 1$. Thus, the string t is viewed as coding the relevant part of an oracle C. By the assumption that $\boldsymbol{P} = \boldsymbol{NP}$, there is a polynomial-time deterministic machine M_2 which accepts the same language as N_2 (where the language is now viewed as a set of pairs (s, t).)

The deterministic machine M^B on s of length n first runs M_1^B to generate the output string t which codes the strings $1^m \in B$ for $m \leq n^k$. Then M^B runs the machine M_2 of the input (s, t). This will produce the same answer as $N^B(s)$. □

At first glance one might expect that showing a similar result for $\boldsymbol{Q}$ to be more complicated since the analogous hypothesis (that $\boldsymbol{NP} \subseteq \boldsymbol{BQP}$) seems weaker. This however poses no problems.

Theorem 3.7. *If $\boldsymbol{Q}$ is Ramsey large, then $\boldsymbol{NP} \not\subseteq \boldsymbol{BQP}$*

Proof. Consider again the set $A = \{1^n : n \in \omega\}$, and suppose that $\boldsymbol{NP} \subseteq \boldsymbol{BQP}$. Just as in Theorem 3.6, we show that for any $B \subseteq A$ that $\boldsymbol{NP}^B \subseteq \boldsymbol{BQP}^B$ which contradicts the hypotheis that $\boldsymbol{Q}$ is Ramsey large. Let N^B be a relativized nondeterministic Turing machine which runs in polynomial time, say bounded by n^k where n is the input length. As in Theorem 3.6, there is a relativized deterministic machine M_1^B which on input s records whether or not $1^m \in B$ for all $m \leq |s|^k$. We can also view M_1^B as a relativized polynomial time quantum machine, since $\boldsymbol{P} \subseteq \boldsymbol{BQP}$. Let N_2 be the unrelativized nondeterministic machine of Theorem 3.6. Since we are assuming $\boldsymbol{NP} \subseteq \boldsymbol{BQP}$, there is an unrelativized polynomial time quantum machine Q which accepts the same language as N_2. As before, by concatenating the machines M_2^B and Q we obtain a $\boldsymbol{BQP}^B$ machine which accepts the same language as N^B.

□

As we mentioned earlier, the analogous statement for $\boldsymbol{S}$ is stronger than the other two. Simply assuming the set $\boldsymbol{S}$ is Ramsey positive is enough to prove $\boldsymbol{PH} \neq \boldsymbol{PSPACE}$. To highlight the difference we make the following definition.

Definition 3.8. An oracle $O \subseteq 2^{<\omega}$ is *tame* if it has at most one string of any given length. We say O is *very tame* if there is a polynomial time computable tame oracle A such that $O \subseteq A$.

In Theorems 3.6 and 3.7, the relevant property of the oracle $\{1^n : n \in \omega\}$ is that is very tame. The point is that relativized $\boldsymbol{NP}^B$ machines are capable of creating lookup tables for tame oracles, whereas relativized $\boldsymbol{P}$ machines are presumably only capable of doing the same for very tame oracles. So, for the class $\boldsymbol{S}$ of oracles separating $\boldsymbol{PSPACE}$ from $\boldsymbol{PH}$ we have the following result.

Theorem 3.9. *If $\boldsymbol{S}$ is Ramsey positive, then $\boldsymbol{PSPACE} \not\subseteq \boldsymbol{PH}$.*

Proof. Suppose that $\boldsymbol{PSPACE} = \boldsymbol{PH}$ and let A be some oracle. It suffices to that there is an oracle $B \subseteq A$ such that $\boldsymbol{PSPACE}^B = \boldsymbol{PH}^B$, as this contradicts $\boldsymbol{S}$ being Ramsey positive. Let $B \subseteq$ be tame, and we show $\boldsymbol{PSPACE}^B = \boldsymbol{PH}^B$. Consider a language $L \in \boldsymbol{PSPACE}^B$, decided by the machine M^B in space n^k. We can have a relativized $\boldsymbol{NP}^B$ machine which on input s creates a lookup table $T_{|s|}$ for strings of length $\leq |s|^k$ by guessing a single string of each length up to $|s|^k$, and querying B to check membership. This is possible since B is tame. There is a nonrelativized $\boldsymbol{PSPACE}$ machine M' which behaved identically to M^B provided it receives the correct lookup table, that is, $M'(T_{|s|}, s) = M^B(s)$ for all strings s. From the hypothesis $\boldsymbol{PSPACE} = \boldsymbol{PH}$ we can get a nonrelativized, $\boldsymbol{PH}$ expression φ which on inputs s and the lookup table T will be equivalent to M', that is $M'(s, T)$ accepts iff $\varphi(s, T)$. The "concatenation" of the $\boldsymbol{NP}$ machine N^B and the $\boldsymbol{PH}$ expression φ can be expressed by a relativized $\boldsymbol{PH}$ expression ψ in which a extra block of existential quantifiers is used to quantify over the existence of the lookup table T as produced by N^B. Then for all strings s we have $\psi^B(s)$ iff $M^B(s)$ accepts, which shows $L \in \boldsymbol{PH}^B$. □

From our results we see that Ramsey largeness is a very strong condition to have on sets of oracle separating two classes. To reflect on why this is, consider an arbitrary oracle A. One might show that two classes relativized to A are different with probability 1, but this says nothing about the kinds of the problems doing the separating. To have that the relevant separating class of oracles be Ramsey large is to say that one can always distill out of an arbitrary racle (by passing to a subset) something which separates them, and that includes starting from oracles which are already very restricted, for example, tame or very tame oracles. To have a separating class of oracles be Ramsey positive is to have that the classes are evidently "different enough" that a tame oracle, despite its limitations, is still powerful enough to create a difference. Similarly, if the separating class of oracles is Ramsey large, then even a very tame oracle can separate the classes.

In the case of $\boldsymbol{PH}$ vs $\boldsymbol{PSPACE}$, the classic separation involving exploiting the lack of small circuits deciding parity requires one to explicitly add multiple strings of the same length at every step. From our results then it seems likely that if one were to find a diagonalization method in which the oracle constructed is specifically tame, then that would lead to a proof that the classes were unequal, via an analogous argument to either Theorem 3.6 or 3.7. Moreover the restrictedness of tame oracles and of very tame oracles seems closely related to the power of $\boldsymbol{NP}$ machines and of $\boldsymbol{P}$ machines, respectively. The condition of being Ramsey positive, in the case again of $\boldsymbol{PH}$ and $\boldsymbol{PSPACE}$, might therefore be considered in another way: that they are so different that a confirmed easy $\boldsymbol{NP}$ computation is enough to confirm separation. Likewise by 3.5 $\boldsymbol{NP}$ is so close to being incomparable to $\boldsymbol{BQP}$ that a confirmed easy $\boldsymbol{P}$ computation could be enough to show full separation. Such information seems potentially valuable in the pursuit of linking our wealth of oracle results with the world of complexity as it actually exists.

Finally, we mention some questions left open from this work. Concerning the Ramsey results we ask:

Question 3.10. Do the converses of Theorems 3.6, 3.7 hold?

In view of Theorems 3.6, 3.7, 3.9 it is natural to ask the following.

Question 3.11. For which other notions of largeness, and for which other pairs of complexity classes does the random oracle hypothesis hold?

References

[1] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, Oct 1997.

[2] Charles H. Bennett and John Gill. Relative to a random oracle A, $\boldsymbol{P}^A \neq \boldsymbol{NP}^A \neq \boldsymbol{co\text{-}NP}^A$. *SIAM J. Comput.*, 10:96–113, 1981.

[3] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[4] Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Håstad, Desh Ranjan, and Pankaj Rohatgi. The random oracle hypothesis is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

[5] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17:13–27, 1984.

[6] Johan Håstad. *Computational Limitations of Small-Depth Circuits*. MIT Press, Cambridge, MA, USA, 1987.

[7] Alenander S. Kechris. *Classical Descriptive Set Theory*. Springer-Verlag, New York, Berlin, 1995.

[8] Dexter Kozen and Michael Machtey. On relative diagonals. Technical Report RC8184, IBM Thomas J. Watson Research Center, April 1980.

[9] Stuart A. Kurtz. On the random oracle hypothesis. *Information and Control*, 57:40–47, 1983.

[10] Adi Shamir. Ip = pspace. *J. ACM*, 39(4):869–877, oct 1992.

[11] Robert Solovay Theodore Baker, John Gill. Relativizations of the $\boldsymbol{P} =? \boldsymbol{NP}$ question. *SIAM Journal on Computing*, December 1975.

[12] A. Yao. Separating the polynomial-time hierarchy by oracles. *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 1–10, 1985.