A Pseudorandum Oracle Characterization of BBP

Jack H. Lutz
Iowa State University, lutz@iastate.edu

Follow this and additional works at: http://lib.dr.iastate.edu/cs_techreports
Part of the Theory and Algorithms Commons

Recommended Citation
http://lib.dr.iastate.edu/cs_techreports/111

This Article is brought to you for free and open access by the Computer Science at Iowa State University Digital Repository. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
A Pseudorandom Oracle Characterization of BBP

TR90-04
Jack H. Lutz
April, 1990
A Pseudorandom Oracle Characterization of BPP*

Jack H. Lutz
Department of Computer Science
Iowa State University
Ames, IA 50011

Abstract

It is known from work of Bennett and Gill and Ambos-Spies that the following conditions are equivalent.

(i) $L \in \text{BPP}$.
(ii) For almost all oracles $A$, $L \in \text{P}^A$.

It is shown here that the following conditions are also equivalent to (i) and (ii).

(iii) The set of oracles $A$ for which $L \in \text{P}^A$ has pspace-measure 1.
(iv) For every pspace-random oracle $A$, $L \in \text{P}^A$.

It follows from this characterization (and its proof) that almost every $A \in \text{ESPACE}$ is $\leq \text{P}^P$-hard for $\text{BPP}^A$. Succinctly, the main content of the proof is that pseudorandom generators exist relative to every pseudorandom oracle.

1 Introduction

The class BPP consists of those decision problems that are feasibly solvable by randomized algorithms. This class, defined by Gill [9], has been shown to admit a variety of equivalent definitions [2, 11, 32, 1, 31, 13, 30, 12, 28, 27]. A particularly elegant and useful characterization of BPP is the following.

**Theorem 1** (Bennett and Gill [2], Ambos-Spies [1]). For a language $L \subseteq \{0,1\}^*$, the following conditions are equivalent.

1. $L \in \text{BPP}$.
2. For almost all oracles $A$, $L \in \text{P}^A$.  

*This research was supported in part by NSF Grants CCR-8809238 and CCR-9157382, and also by DIMACS, where the author was visiting while part of this work was performed.
The “almost all” in condition (2) here refers to Lebesgue measure on the set of all oracles. (Oracles in this paper are languages $A \subseteq \{0,1\}^\ast$.) That is, if an oracle $A$ is chosen probabilistically, using an independent toss of a fair coin to decide whether each string $x \in \{0,1\}^\ast$ is in $A$, then condition (2) asserts that $L \in \mathbb{P}^A$ with probability one.

Interesting though it is, this characterization demands a more careful analysis. Since BPP is countable, Theorem 1 implies that almost every oracle is $\leq^P_T$-hard for BPP. Nevertheless, Theorem 1 gives no information regarding which oracles are $\leq^P_T$-hard for BPP. (The inclusion BPP $\subseteq \Sigma^P_2 \cap \Pi^P_2$ of Sipser and Gács [25] implies that oracles that are $\leq^P_T$-hard for $\Sigma^P_2 \cap \Pi^P_2$ have this property, but by Theorem 1 this is only a measure 0 set of oracles, unless BPP $= \Sigma^P_2 \cap \Pi^P_2$.)

In this paper we refine Theorem 1 by proving the following.

**Main Theorem.** For a language $L \subseteq \{0,1\}^\ast$, the following conditions are equivalent.

1. $L \in \text{BPP}$.
2. The set of oracles $A$ for which $L \in \mathbb{P}^A$ has pspace-measure 1.
3. For every pspace-random oracle $A$, $L \in \mathbb{P}^A$.

(Conditions (2) and (3) here refer to the resource-bounded measure theory and measure-theoretic pseudorandomness of Lutz [18]; see §3 below for details.)

Intuitively, the Main Theorem says that every sufficiently random oracle is $\leq^P_T$-hard for BPP, and that pspace-randomness is sufficient here. Of course every random oracle (i.e., every language whose characteristic sequence is algorithmically random in the equivalent senses of Martin-Löf [20], Levin [14], Schnorr [23], Chaitin [5, 6], Solovay [26], and Shen’ [24]) is pspace-random, so it follows immediately from the Main Theorem that every random oracle is $\leq^P_T$-hard for BPP. Since almost every oracle is random [20], this in turn gives the $(1) \implies (2)$ part of Theorem 1. However, the Main Theorem is much stronger than this. For example, since every pspace-measure 1 set has measure 1 in $\text{SPACE} = \text{DSP}^{2\text{linear}}$ [18], the $(1) \implies (2)$ part of the Main Theorem tells us that for each $L \in \text{BPP}$, $L$ is $\leq^P_T$-reducible to almost every $A \in \text{SPACE}$. Similarly, since almost every language in $\text{E}_2\text{SPACE} = \text{DSP}^{2\text{polynomial}}$ is pspace-random [18], the $(1) \implies (3)$ part of the Main Theorem tells us that almost every language in $\text{E}_2\text{SPACE}$ is $\leq^P_T$-hard for BPP. In fact, our proof tells us more, namely that almost every language $A \in \text{SPACE}$ is $\leq^P_T$-hard for BPP.

2 Overview of Proof

The following notion of hardness relative to oracle circuits is central to the proof of the Main Theorem.

**Definition** (Nisan and Wigderson [21, 22]). Given languages $L, A \subseteq \{0,1\}^\ast$, a real $\delta > 0$, and $n, s \in \mathbb{N}$, $L$ is $(\delta, s)^A$-hard at $n$ if

$$|L(\gamma^A) \triangle L_{sn}| > 2^{n-1} (1 - \delta)$$

for every $n$-input oracle circuit $\gamma$ with size$(\gamma) \leq s$. (Here $L_{sn}$ denotes $L \cap \{0,1\}^n$.) The hardness of $L$ relative to $A$ is the function $H^A_L : \mathbb{N} \to \mathbb{N}$ defined by

$$H^A_L(n) = \max\{h \in \mathbb{N} \mid L \text{ is } (h^{-1}, h)^A\text{-hard at } n\}.$$
(See [29] or [19] for details concerning oracle circuits.)

Thus a language \( L \) is \((\delta, s)^A\)-hard at \( n \) if \( \gamma^A \) computes \( L \) incorrectly on at least \( 50(1 - \delta) \) percent of the inputs in \( \{0, 1\}^n \), whenever \( \gamma \) is an \( n \)-input oracle circuit of size \( s \).

For each real \( 0 < \alpha \leq 1 \) and each oracle \( A \subseteq \{0, 1\}^* \), define the relativized hardness class

\[
H^A_\alpha = \{ L \subseteq \{0, 1\}^* | H^A_L(n) > 2^{\alpha n} \text{ a.e.} \}
\]

(We say that a condition \( \Theta(n) \) holds almost everywhere (a.e.) if it holds for all but finitely many \( n \in \mathbb{N} \). We say that \( \Theta(n) \) holds infinitely often (i.o.) if it holds for infinitely many \( n \in \mathbb{N} \).) Also define

\[
H^4 = \bigcap_{0 < \alpha \leq \frac{1}{2}} H^A_\alpha.
\]

If \( E^A = \text{DTIME}^A(2^{\text{linear}}) \) contains a hard language, then this language can be used to construct a pseudorandom bit generator that is quick enough and secure enough to achieve \( \text{P}^A = \text{BPP}^A \). That is, we have the following.

**Theorem 2** (Nisan and Wigderson [21, 22]). For every oracle \( A \) and every \( 0 < \alpha < 1 \), if \( E^A \cap H^4 \neq \emptyset \), then \( \text{P}^A = \text{BPP}^A \). \( \square \)

The proof of Theorem 2, a relativization of arguments in [21, 22], will not be given here. The following result, which is the main technical content of this paper, will be proven in §4.

**Theorem 3.** \( \mu_{\text{pspace}}(\{ A \mid E^A \cap H^4 \neq \emptyset \}) = 1 \). (That is, the indicated set of oracles has pspace-measure 1.)

**Corollary 4.** \( \mu_{\text{pspace}}(\{ A \mid \text{P}^A = \text{BPP}^A \}) = 1 \). \( \square \)

The proof of the Main Theorem is now easy. If (1) holds, then \( \text{P}^A = \text{BPP}^A \) implies \( L \in \text{P}^A \), so (2) follows by Corollary 4. If (2) holds, then (3) holds because every pspace-random language is, by definition, an element of every pspace-measure 1 set [18]. Finally, almost every oracle \( A \) is pspace-random [18], so (1) follows from (3) by the \( (2) \implies (1) \) part of Theorem 1.

\( \square \)

The relationship between pseudorandom generators and pseudorandom oracles is a particularly interesting aspect of this proof. A pseudorandom generator is a function \( G : \{0, 1\}^* \times \mathbb{N} \to \{0, 1\}^* \) such that \( |G(x, n)| = n \) for all \( x \) and \( n \). Given a function \( l : \mathbb{N} \to \mathbb{N} \) and an oracle \( A \subseteq \{0, 1\}^* \), a generator \( G \) is \( A \)-quick and \( A \)-secure on seeds of length \( l \), and we write \( G : l \xrightarrow{A} n \), if (i) \( G(x, n) \) is deterministically computable in \( 2^{O(l)} \) time relative to \( A \) whenever \( |x| = l(n) \), and (ii) for every family \( \gamma = (\gamma_n) \) of oracle circuits with size\((\gamma_n) = O(n)\), we have

\[
\left| \Pr \left[ \gamma_n^A(G(x, n)) = 1 \right] - \Pr \left[ \gamma_n^A(y) = 1 \right] \right| < \frac{1}{n} \text{ a.e.,}
\]

where \( x \in \{0, 1\}^{l(n)} \) and \( y \in \{0, 1\}^n \) are chosen according to the uniform distributions.
The main part of Nisan and Wigderson’s proof of Theorem 2 shows that for every real
$0 < \alpha < 1$, there exists $c \in \mathbb{N}$ such that for all $A \subseteq \{0,1\}^*$, if $E^A \cap H^A \neq \emptyset$, then there is a
generator $G : c \log n \xrightarrow{A} n$. Putting this together with Theorem 3 gives the following.

**Theorem 5.** There is a positive integer $c$ such that, for every pspace-random oracle
$A \subseteq \{0,1\}^*$, there exists a pseudorandom generator $G : c \log n \xrightarrow{A} n$. \(\square\)

Less formally, this says that pseudorandom generators exist relative to every pseudorandom oracle.

## 3 pspace-Measure and pspace-Randomness

In this section we review some fundamentals of resource-bounded measure and pseudorandomness, where the resource bound is polynomial space. For more details, examples, and proofs, see [18].

We work two alphabets, the binary alphabet $\{0,1\}$ and the extended binary alphabet
$\Sigma = \{0,1,\bot\}$. The symbol $\bot$ (“bottom”) denotes an “undefined bit.” We fix the partial
ordering $\sqsubseteq$ of $\Sigma$ in which $\bot \sqsubseteq 0, \bot \sqsubseteq 1$, and $0$ and $1$ are incomparable. Given a string or
sequence $x \in \Sigma^* \cup \Sigma^\infty$, we write $x[i]$ for the $i^{th}$ bit of $x$ and $x[i..j]$ for the string consisting
of the $i^{th}$ through $j^{th}$ bits of $x$. We also fix the standard enumeration $s_0 = \lambda, s_1 = 0, s_2 = 1, s_3 = 00, \ldots$ of $\{0,1\}^*$, and write $x[w] = x[i]$ whenever $w = s_i$ and $0 \leq i < |x|$. We extend
$\sqsubseteq$ bitwise to strings and sequences, i.e., $x \sqsubseteq y$ iff $(\forall i \in \mathbb{N})x'[i] \sqsubseteq y'[i]$, where $x' = x$ if
$|x| = \infty$, $x' = x \bot \infty$ if $|x| < \infty$, and $y'$ is defined similarly. The cylinder specified by a string
$x \in \Sigma^*$ is $C_x = \{A \subseteq \{0,1\}^* \mid x \sqsubseteq \chi_A\}$, where $\chi_A \in \{0,1\}^\infty$ is the characteristic sequence
of $A$. i.e., each $\chi_A[i]$ is 1 if $s_i \in A$ and 0 otherwise. We use the symbol $\top$ (“top”) to specify the empty set, i.e., $C_\top = \emptyset$. For $x, y \in \Sigma^*$, we let $x \wedge y$ be the shortest string such that
$C_{x \wedge y} = C_x \cap C_y$. Note that $x \wedge y = \top$ if $x$ and $y$ are incompatible, i.e., if $C_x \cap C_y = \emptyset$. The measure $\mu(x)$ of a cylinder $C_x$ is the probability that $A \in C_x$ when $A \subseteq \{0,1\}^*$ is chosen according to the random experiment in which an independent toss of a fair coin is used to decide whether each string $w \in \{0,1\}^*$ is in $A$. Thus if we let $\#(b,x)$ denote the number of occurrences of the symbol $b$ in the string $x$ and define

$$\|x\| = \begin{cases} \#(0,x) + \#(1,x) & \text{if } x \in \Sigma^* \\ \infty & \text{if } x = \top, \end{cases}$$

then $\mu(x) = 2^{-\|x\|}$ for all $x \in \Sigma^* \cup \{\top\}$.

We fix once and for all a one-to-one pairing function $\langle,\rangle$ from $\{0,1\}^* \times \{0,1\}^*$ onto $\{0,1\}^*$ such that the pairing function and its associated projections, $\langle x,y \rangle \mapsto x$ and $\langle x,y \rangle \mapsto y$ are computable in polynomial time. We insist further that this pairing function satisfy the following condition for all $x, y \in \{0,1\}^*$: $\langle x,y \rangle \in \{0\}^*$ if and only if $x,y \in \{0\}^*$. This condition canonically induces a pairing function $\langle,\rangle$ from $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$. We write $(x,y,z)$ for $\langle x,\langle y,z \rangle \rangle$, etc., so that tuples of any fixed length are coded by the pairing function.

We let $D = \{m2^{-n} \mid m,n \in \mathbb{N}\}$ be the set of nonnegative dyadic rationals. Many functions in this paper take their values in $D$ or in $[0,\infty)$, the set of nonnegative real numbers. In fact, with the exception of some functions that map into $[0,\infty)$, our functions
are of the form \( f : X \to Y \), where each of the sets \( X, Y \) is \( \mathbb{N}, \{0,1\}^*, \mathbb{D} \), or some cartesian product of these sets. Formally, in order to have uniform criteria for their computational complexities, we regard all such functions as mapping \( \{0,1\}^* \) into \( \{0,1\}^* \). For example, a function \( f : \mathbb{N}^2 \times \{0,1\}^* \to \mathbb{N} \times \mathbb{D} \) is formally interpreted as a function \( f : \{0,1\}^* \to \{0,1\}^* \).

Under this interpretation, \( f(i, j, w) = (k, q) \) means that \( f(\langle 0^i, \langle 0^j, w \rangle \rangle) = \langle 0^k, \langle u, v \rangle \rangle \), where \( u \) and \( v \) are the binary representations of the integer and fractional parts of \( q \), respectively. Moreover, we only care about the values of \( f \) for arguments of the form \( \langle 0^i, \langle 0^j, w \rangle \rangle \), and we insist that these values have the form \( \langle 0^k, \langle u, v \rangle \rangle \) for such arguments.

For a function \( f : \mathbb{N} \times X \to Y \) and \( k \in \mathbb{N} \), we define the function \( f_k : X \to Y \) by \( f_k(x) = f(\langle 0^k, x \rangle) \). We then regard \( f \) as a “uniform enumeration” of the functions \( f_0, f_1, f_2, \ldots \). For a function \( f : \mathbb{N}^n \times X \to Y \) (\( n \geq 2 \)), we write \( f_{k,i} = (f_k)_i \), etc. For a function \( f : \{0,1\}^* \to \{0,1\}^* \), we write \( f^n \) for the \( n \)-fold composition of \( f \) with itself.

We work with the resource bound

\[ \text{pspace} = \{ f : \{0,1\}^* \to \{0,1\}^* \mid f \text{ is computable in polynomial space} \}. \]

(The length \( |f(x)| \) of the output is included as part of the space used in computing \( f \).)

Resource-bounded measure and pseudorandomness were originally developed in terms of “modulated covering by cylinders” [16, 17, 15]. Though the main results of these papers are true, the underlying development was technically flawed. This situation was remedied in [18], where resource-bounded measure was reformulated in terms of density functions. We review relevant aspects of the latter formulation here.

A density function is a function \( d : \{0,1\}^* \to [0, \infty) \) satisfying

\[ d(x) \geq \frac{d(x0) + d(x1)}{2} \]

for all \( x \in \{0,1\}^* \). The global value of a density function \( d \) is \( d(\lambda) \). An \( n \)-dimensional density system (n-DS) is a function \( d : \mathbb{N}^n \times \{0,1\}^* \to [0, \infty) \) such that \( d_k \) is a density function for every \( k \in \mathbb{N}^n \). It is sometimes convenient to regard a density function as a 0-DS.

A computation of an n-DS \( d \) is a function \( \hat{d} : \mathbb{N}^{n+1} \times \{0,1\}^* \to \mathbb{D} \) such that

\[ \left| \hat{d}_{k,r}(x) - d_k(x) \right| \leq 2^{-r} \quad (3.1) \]

for all \( k \in \mathbb{N}^n, r \in \mathbb{N}, \) and \( x \in \{0,1\}^* \). A pspace-computation of an n-DS \( d \) is a computation \( \hat{d} \) such that \( \hat{d} \in \text{pspace} \). An n-DS is pspace-computable if there exists a pspace-computation \( \hat{d} \) of \( d \). (Note that (3.1) implies that

\[ d_k(x) = \lim_{r \to \infty} \hat{d}_{k,r}(x) \]

for all \( k \in \mathbb{N}^n \) and \( x \in \{0,1\}^* \).)

The set covered by a density function \( d \) is

\[ S[d] = \bigcup_{x \in \{0,1\}^* \land d(x) \geq 1} C_x. \]

A density function \( d \) covers a set \( X \) of languages if \( X \subseteq S[d] \). A null cover of a set \( X \) of languages is a 1-DS \( d \) such that, for all \( k \in \mathbb{N} \), \( d_k \) covers \( X \) with global value \( d_k(\lambda) \leq 2^{-k} \).
It is easy to show [18] that a set $X$ of languages has classical Lebesgue measure 0 (i.e., probability 0 in the coin-tossing random experiment) if and only if there exists a null cover of $X$. In this paper we are interested in the situation where the null cover $d$ is pspace-computable.

**Definitions.** Let $X$ be a set of languages and let $X^c$ denote the complement of $X$.

1. A pspace-null cover of $X$ is a null cover of $X$ that is pspace-computable.
2. $X$ has pspace-measure 0, and we write $\mu_{\text{pspace}}(X) = 0$, if there exists a pspace-null cover of $X$.
3. $X$ has pspace-measure 1, and we write $\mu_{\text{pspace}}(X) = 1$, if $\mu_{\text{pspace}}(X^c) = 0$.
4. $X$ has measure 0 in $\text{ESPACE} = DSPACE(2^{\text{linear}})$, and we write $\mu(X \mid \text{ESPACE}) = 0$, if $\mu_{\text{pspace}}(X \cap \text{ESPACE}) = 0$.
5. $X$ has measure 1 in $\text{ESPACE}$, and we write $\mu(X \mid \text{ESPACE}) = 1$, if $\mu(X^c \mid \text{ESPACE}) = 0$. In this case, we say that $X$ contains almost every language in $\text{ESPACE}$.

It is shown in [18] that these definitions endow $\text{ESPACE}$ with internal measure-theoretic structure. Specifically, if $\mathcal{I}$ is either the collection $\mathcal{I}_{\text{pspace}}$ of all pspace-measure 0 sets or the collection $\mathcal{I}_{\text{ESPACE}}$ of all sets of measure 0 in $\text{ESPACE}$, then $\mathcal{I}$ is a “pspace-ideal,” i.e., is closed under subsets, finite unions, and “pspace-unions” (countable unions that can be generated in polynomial space). More importantly, it is shown that the ideal $\mathcal{I}_{\text{ESPACE}}$ is a proper ideal, i.e., that $\text{ESPACE}$ does not have measure 0 in $\text{ESPACE}$.

Our proof of Theorem 3 does not proceed directly from the above definitions. Instead we use a sufficient condition, proved in [18], for a set to have pspace-measure 0. To state this condition we need a polynomial notion of convergence for infinite series. All our series here consist of nonnegative terms. A modulus for a series $\sum_{n=0}^{\infty} a_n$ is a function $m : \mathbb{N} \to \mathbb{N}$ such that

$$\sum_{n=m(j)}^{\infty} a_n \leq 2^{-j}$$

for all $j \in \mathbb{N}$. A series is p-convergent if it has a modulus that is a polynomial. A sequence

$$\sum_{k=0}^{\infty} a_{j,k} \quad (j = 0, 1, 2, \ldots)$$

of series is uniformly p-convergent if there exists a polynomial $m : \mathbb{N}^2 \to \mathbb{N}$ such that, for each $j \in \mathbb{N}$, $m_j$ is a modulus for the series $\sum_{k=0}^{\infty} a_{j,k}$. We will use the following sufficient condition for uniform p-convergence. (This well-known lemma is easily verified by routine calculus.)

**Lemma 6.** Let $a_{j,k} \in [0, \infty)$ for all $j, k \in \mathbb{N}$. If there exist a real $\varepsilon > 0$ and a polynomial $g : \mathbb{N} \to \mathbb{N}$ such that $a_{j,k} \leq e^{-k^\varepsilon}$ for all $j, k \in \mathbb{N}$ with $k \geq g(j)$, then the series

$$\sum_{k=0}^{\infty} a_{j,k} \quad (j = 0, 1, 2, \ldots)$$

6
are uniformly p-convergent. □

The proof of Theorem 3 is greatly simplified by using the following special case (for \text{pspace}) of a uniform, resource-bounded generalization of the classical first Borel-Cantelli lemma.

**Lemma 7** (Borel [3], Cantelli [4], Lutz [18]). If \( d \) is a \text{pspace}-computable 2-DS such that the series
\[
\sum_{k=0}^{\infty} d_{j,k}(\lambda) \quad (j = 0, 1, 2, \ldots)
\]
are uniformly p-convergent, then
\[
\mu_{\text{pspace}} \left( \bigcup_{j=0}^{\infty} \bigcap_{t=0}^{\infty} \bigcup_{k=t}^{\infty} S[d_{j,k}] \right) = 0.
\]

□

If we write \( S_j = \bigcap_{t=0}^{\infty} \bigcup_{k=t}^{\infty} S[d_{j,k}] \) and \( S = \bigcup_{j=0}^{\infty} S_j \), then Lemma 7 gives a sufficient condition for concluding that \( S \) has \text{pspace}-measure 0. Note that each \( S_j \) consists of those languages \( A \) that are in infinitely many of the sets \( S[d_{j,k}] \).

Finally, we review the notion of \text{pspace}-randomness. A \text{pspace}-test is a set \( X \) of languages such that \( \mu_{\text{pspace}}(X) = 1 \). A language \( A \) passes a \text{pspace}-test \( X \) if \( A \in X \). A language \( A \) is \text{pspace}-random, and we write \( A \in \text{RAND}(\text{pspace}) \), if \( A \) passes all \text{pspace}-tests. That is,
\[
\text{RAND}(\text{pspace}) = \bigcap_{\mu_{\text{pspace}}(X) = 1} X.
\]

Since every finite subset of \text{ESPACE} has \text{pspace}-measure 0 \cite{18}, it is immediate that
\[
\text{RAND}(\text{pspace}) \cap \text{ESPACE} = \emptyset. \quad (3.2)
\]

Moreover, every \text{pspace}-random language has essentially maximum circuit-size complexity and space-bounded Kolmogorov complexity \cite{18}. Intuitively, \text{pspace}-random languages are “random enough for all \text{pspace}-computable purposes.” On the other hand, \text{pspace}-random languages may be computable. In fact, notwithstanding (3.2), almost every language in \( \text{E}_2\text{SPACE} = \text{DSPACE}(2^{\text{polynomial}}) \) is \text{pspace}-random \cite{18}.

## 4 Hardness Under Pseudorandom Oracles

In this section we prove Theorem 3. For each \( A \subseteq \{0,1\}^* \), let
\[
\text{ODD}(A) = \{ u \in \{0,1\}^* \mid |C(u, A)| \text{ is odd} \},
\]
where
\[
C(u, A) = \{ uv \in A \mid |v| = 2|u| \},
\]
and let
\[
X = \{ A \mid \text{ODD}(A) \not\subseteq H^4 \}.
\]
Then \( \text{ODD}(A) \in \text{E}^d \) for all \( A \), so it suffices to prove that \( \mu_{\text{pspace}}(X) = 0. \)

7
For each $j, k \in \mathbb{N}$, let

$$X_{j,k} = \begin{cases} \{ A \mid H_{\text{ODD}(A)}^A(n) \leq 2^{\alpha(l)n} \} & \text{if } j = 2^j \text{ and } k = 2^n \\ \emptyset & \text{if } j \text{ and } k \text{ are not of this form,} \end{cases}$$

where $\alpha(l) = \frac{l+1}{3l+1}$. (Note that $\alpha(0) = \frac{1}{4}$, $\alpha(l)$ is strictly increasing, and $\lim_{l \to \infty} \alpha(l) = \frac{1}{3}$.) It is clear that

$$X = \bigcup_{j=0}^{\infty} \bigcap_{i=0}^{\infty} \bigcup_{k=i}^{j} X_{j,k}. \quad (4.1)$$

We will use (4.1) and Lemma 7 to prove that $\mu_{\text{pspace}}(X) = 0$.

For all $l, n \in \mathbb{N}$, let $j = 2^j$, $k = 2^n$, and define the sets

$$\text{OCIRC}(2^{\alpha(l)n}) = \{ \gamma \mid \gamma \text{ is a novel n-input oracle circuit with size(\gamma) \leq k^{\alpha(l)}}, \}
\quad \text{DELTA}(l, n) = \{ D \subseteq \{0,1\}^n \mid |D| \leq \frac{k}{2}(1 - k^{-\alpha(l)}) \}.$$

(An $n$-input oracle circuit $\gamma$ is novel if it is functionally distinct from all those preceding it in a standard enumeration.) For all $\gamma \in \text{OCIRC}(2^{\alpha(l)n})$ and $D \in \text{DELTA}(l, n)$, then, let

$$Y_{\gamma,D} = \{ A \mid L^A(\gamma) \triangle D = \text{ODD}(A)_{=n} \}.$$

Note that

$$X_{j,k} = \bigcup_{\gamma \in \text{OCIRC}(2^{\alpha(l)n})} \bigcup_{D \in \text{DELTA}(l, n)} Y_{\gamma,D}. \quad (4.2)$$

for all $l, n \in \mathbb{N}$, where $j = 2^j$ and $k = 2^n$.

Define $d : \mathbb{N}^2 \times \{0,1\}^* \to [0, \infty)$ by

$$d_{j,k}(x) = \begin{cases} \sum_{\gamma \in \text{OCIRC}(2^{\alpha(l)n})} \sum_{D \in \text{DELTA}(l, n)} P(Y_{\gamma,D} \mid C_x) & \text{if } j = 2^j \text{ and } k = 2^n \\ 0 & \text{if } j \text{ and } k \text{ are not of this form.} \end{cases} \quad (4.3)$$

The conditional probability

$$P(Y_{\gamma,D} \mid C_x) = \Pr[A \in Y_{\gamma,D} \mid A \in C_x]$$

in (4.3) is computed according to the uniform distribution on languages $A \subseteq \{0,1\}^*$, i.e., the random experiment in which $A$ is chosen probabilistically, using an independent toss of a fair coin to decide whether each string $y \in \{0,1\}$ is in $A$. Note that $P(X_{j,k} \mid C_x) \leq d_{j,k}(x)$ for all $j, k \in \mathbb{N}$ and $x \in \{0,1\}^*$. (This inequality may be strict because the union (4.2) is not a disjoint union.)

By (4.1) and Lemma 7, it suffices to prove the following three claims.

**Claim 1.** $d$ is a pspace-computable 2-DS.

**Claim 2.** For all $j, k \in \mathbb{N}$, $X_{j,k} \subseteq S[d_{j,k}]$. 

8
Claim 3. The series
\[ \sum_{k=0}^{\infty} d_{j,k}(\lambda) \quad (j = 0, 1, 2, \ldots) \]
are uniformly p-convergent.

To prove Claim 1, first note that each
\[
P(Y_{\gamma,D} \mid C_x) = \frac{P(Y_{\gamma,D} \cap C_x)}{P(C_x)}
= \frac{P(Y_{\gamma,D} \cap C_{x0}) + P(Y_{\gamma,D} \cap C_{x1})}{P(C_x)}
= \frac{P(Y_{\gamma,D} \cap C_{x0})}{2P(C_{x0})} + \frac{P(Y_{\gamma,D} \cap C_{x1})}{2P(C_{x1})}
= \frac{P(Y_{\gamma,D} \mid C_{x0}) + P(Y_{\gamma,D} \mid C_{x1})}{2},
\]
so
\[
d_{j,k}(x) = \frac{d_{j,k}(x0) + d_{j,k}(x1)}{2}
\]
for all \( j, k \in \mathbb{N} \) and \( x \in \{0, 1\}^* \). It follows that \( d \) is a 2-DS.

It is clear that we can use (4.3) to compute \( d \), provided that we can compute the conditional probabilities \( P(Y_{\gamma,D} \mid C_x) \). We thus focus on this computation.

Fix \( \gamma \in \text{OCIRC}(2^{a(l)n}) \) and \( D \in \text{DELTA}(l, n) \). Let \( \text{SOURCES}(n) = \{0, 1\}^{k^3+k^2} \). For each \( z \in \text{SOURCES}(n) \), let a string \( w \in \Sigma^* \) of length \( 2^{k^3+k^2} - 1 \) and a set \( \text{ODD} \subseteq \{0, 1\}^n \) be constructed as follows. (For each \( A \in Y_{\gamma,D} \), this process will, for some \( z \), produce a string \( w \subseteq \chi_A \) and corresponding set \( \text{ODD} = \text{ODD}(A) \cap \{0, 1\}^n \).) Initially, \( \text{ODD} = \emptyset \) and \( w \) is all \( \perp \)'s. Then simulate \( \gamma \) on the successive inputs \( u \in \{0, 1\}^* \). Each time \( \gamma \) queries a string \( y \) in this simulation, do if \( w[y] = \perp \) then \( (w[y], z) := (\text{head}(z), \text{tail}(z)) \). (Note that \( |w| \) has been chosen large enough for \( w[y] \) to exist here.) Then, in any case, use \( w[y] \) as the response to the query. If \( \gamma(u) = 1 \) in this simulation, do \( \text{ODD} := \text{ODD} \cup \{u\} \). After \( \gamma \) has been simulated on all inputs, do \( \text{ODD} := \text{ODD} \triangle D \). At this point, note that at most \( k^{1+a(l)} < k^{3/2} \) of the bits \( w[y] \) of \( w \) are in \( \{0, 1\} \); the rest are still \( \perp \). Finally, use the remaining bits of \( z \) (actually a portion of them, as needed) to complete the specification of \( w \) as follows. For each \( u \in \{0, 1\}^n \), first use bits of \( z \) to fill in all but one of the values \( w[uv] \) for \( v \in \{0, 1\}^{2n} \); then define the remaining bit \( w[uv] \) according to whether \( u \in \text{ODD} \). (The measure argument in Claim 3 below works precisely because these \( k \) bits—one for each \( u \)—are determined by \( \text{ODD} \).) Finally, let \( z' \) be the initial segment of the original string \( z \in \text{SOURCES}(n) \) consisting of those bits actually used in this construction. Note that \( |z'| < k^3 + k^{3/2} \) and that all but \( |z'| \) of \( w \) are still \( \perp \). Since \( w \) depends only upon the prefix \( z' \) of \( z \), we write \( w = w(z') \).

Let \( \text{SOURCES}'(n) = \{z' \mid z \in \text{SOURCES}(n)\} \). Since \( \gamma \) is a fixed oracle circuit (whose gates we simulate in a fixed topological order), we have \( C_{w(z_1)} \cap C_{w(z_2)} = \emptyset \) for distinct \( z_1, z_2 \in \text{SOURCES}'(n) \). Moreover, it is clear that
\[
Y_{\gamma,D} = \bigcup_{z' \in \text{SOURCES}'(n)} C_{w(z')}.
\]
It follows that, for all \( x \in \{0,1\}^* \),

\[
P(Y_{\gamma,D} \mid C_x) = \sum_{z' \in \text{SOURCES}'(n)} P(C_{w(z')} \mid C_x)
= \sum_{z' \in \text{SOURCES}'(n)} 2^{-\|x \wedge w(z')\|}.
\]

(4.5)

This is the basis for our computation. Given \( j, k, x, \gamma, D \), and \( z' \), it is clear that we can compute \( 2^{-\|x \wedge w(z')\|} \) in space polynomial in \( j + k + |x| \). (The string \( w(z') \) has fewer than \( k^3 + k^2 + k \) non-\( \perp \) bits, so it can be stored in space polynomial in \( k \).) We can find the successive strings \( z' \in \text{SOURCES}'(n) \) by a depth-first search of \( \{0,1\}^{k^3+k^2} \), also in polynomial space. We can thus use (4.5) to calculate \( P(Y_{\gamma,D} \mid C_x) \) in space polynomial in \( j + k + |x| \). As already noted, we can then use (4.3) to calculate \( d_{j,k}(x) \) in polynomial space. This proves that \( d \in \text{pspace} \), whence \( d \) is certainly pspace-computable, affirming Claim 1.

To prove Claim 2, fix \( j, k \in \mathbb{N} \). If \( j \) and \( k \) are not of the form \( j = 2^j \) and \( k = 2^k \), then \( X_{j,k} \subseteq S[d_{j,k}] \) holds trivially. If \( j = 2^j \) and \( k = 2^k \), let \( A \in X_{j,k} \). By (4.2), fix \( \gamma \in \text{OCIRC}(2^{a(1)n}) \) and \( D \in \text{DELTA}(l, n) \) such that \( A \in Y_{\gamma,D} \). By (4.4), fix \( z' \in \text{SOURCES}'(n) \) such that \( A \in C_{w(z')} \). Let \( m = |w(z')| \). Then

\[
d_{j,k}(\chi_A[0..,m-1]) \geq P(Y_{\gamma,D} \mid C_{\chi_A[0..,m-1]})
= P(C_{w(z')} \mid C_{\chi_A[0..,m-1]})
= 1,
\]

so \( A \in S[d_{j,k}] \) in any case. This proves Claim 2.

To prove Claim 3, we estimate the global values \( d_{j,k}(\lambda) \). Fix \( l, n \in \mathbb{N} \) and let \( j = 2^j, k = 2^k \). Fix \( \gamma \in \text{OCIRC}(2^{a(1)n}) \) and \( D \in \text{DELTA}(l, n) \). By (4.5) and the fact that \( k \) bits of each \( w(z') \) are determined by \( \text{ODD} \), we have

\[
P(Y_{\gamma,D} \mid C_\lambda) = \sum_{z' \in \text{SOURCES}'(n)} 2^{-\|w(z')\|}
= 2^{-k} \sum_{z' \in \text{SOURCES}'(n)} 2^{-|z'|}
= 2^{-k}.
\]

(The last equality here holds because every string \( z \in \text{SOURCES}(n) \) has exactly one prefix \( z' \in \text{SOURCES}'(n) \).) Since \( \gamma \) and \( D \) are arbitrary here, it follows by (4.3) that

\[
d_{j,k}(\lambda) \leq |\text{OCIRC}(2^{a(1)n})| \cdot |\text{DELTA}(l, n)| \cdot 2^{-k}.
\]

(4.6)

A routine counting argument shows that

\[
|\text{OCIRC}(2^{a(1)n})| \leq a(4ck^a(1))k^{a(1)},
\]

where \( a = 2685 \). (This is Lemma 4.2 of [19].) It follows that there is a constant \( n_1 \in \mathbb{N} \) such that

\[
|\text{OCIRC}(2^{a(1)n})| \leq 2^{k^{a(1)\log k}}
\]

(4.7)
for all $l, n \in \mathbb{N}$ with $n \geq n_1$. (The constant $n_1$ does not depend upon $l$ here because $\alpha(l) < \frac{1}{3}$ for all $l$.) By the Chernoff bound (see [7, 8, 10]),

$$|\Delta(l, n)| \leq 2^k \rho^k,$$  \hspace{1cm} (4.8)

where

$$\rho = \left(1 - \varepsilon^2\right)^{-\frac{1}{2}} \left(\frac{1 - \varepsilon}{1 + \varepsilon}\right)^{\varepsilon}, \quad \varepsilon = k^{-\alpha(l)}. \hspace{1cm} (4.9)$$

Calculating with Taylor approximations, we have

$$\left(\frac{1 - \varepsilon}{1 + \varepsilon}\right)^{\varepsilon} = (1 - 2\varepsilon + o(\varepsilon))^\varepsilon = \varepsilon^{\varepsilon \ln(1 - 2\varepsilon + o(\varepsilon))} = e^{-2\varepsilon^2 + o(\varepsilon^2)} \approx 1 - 2\varepsilon^2 + o(\varepsilon^2)$$

as $\varepsilon \to 0$. Since $(1 - \varepsilon^2)(1 - \frac{1}{2}\varepsilon^2) = 1 - \frac{3}{2}\varepsilon^2 + o(\varepsilon^2)$ as $\varepsilon \to 0$, it follows that

$$\left(\frac{1 - \varepsilon}{1 + \varepsilon}\right)^{\varepsilon} < (1 - \frac{3}{2}\varepsilon^2)(1 - \frac{1}{2}\varepsilon^2) \quad \hspace{1cm} (4.10)$$

for all sufficiently small $\varepsilon$. By (4.8), (4.9), and (4.10), there is a constant $n_2 \in \mathbb{N}$ such that

$$|\Delta(l, n)| \leq 2^k \left(1 - \frac{\varepsilon^2}{2}\right)^\frac{k}{2} = 2^k \frac{1}{2} \log \left(1 - \frac{\varepsilon^2}{2}\right) \leq 2^{k - c_k\varepsilon^2} \leq 2^{k - c_k^{-2\alpha(l)}} \quad \hspace{1cm} (4.11)$$

for all $l, n \in \mathbb{N}$ with $n \geq n_2$, where $c = \frac{1}{4\ln 2}$. (The constant $n_2$ does not depend upon $l$ because $\varepsilon = k^{-\alpha(l)} \leq k^{-\alpha(0)} = k^{-\frac{1}{2}}$ in any case.)

Let $k_0 = 2^{\max\{n_1, n_2\}}$. By (4.3), (4.6), (4.7), and (4.11), we have

$$d_{j,k}(\lambda) \leq 2^{k_{\alpha(1)} \log k - c_k^{-2\alpha(l)}} \quad \hspace{1cm} (4.12)$$

for all $j, k \in \mathbb{N}$ with $j = 2^2 l$ and $k \geq k_0$. Define a polynomial $g: \mathbb{N} \to \mathbb{N}$ by

$$g(j) = 2^{185j^{381}} + k_0$$

for all $j \in \mathbb{N}$. Writing $t = \ln k$ and $a = 3l + 4$, we have

$$k \geq g(j) \quad \implies \quad t \geq 185 \ln 2 + 381l^2 \ln 2 \geq 128 + 264l^2 \geq 8a^2. \quad \hspace{1cm} (4.13)$$

Examining the function $f(t) = e^t - 4t - 4$ and its derivative shows that $f(t) > 0$ for all $t \geq 8a^2$. By (4.13), then,

$$k \geq g(j) \quad \implies \quad e^\frac{t}{2} - 4t - 4 \geq 0 \quad \implies \quad k^\frac{t}{2} - 4 \ln k - 4 \geq 0 \quad \implies \quad c k^{1 - 3\alpha(l)} - \log k - \log e \geq 0 \quad \implies \quad 2^{k_{\alpha(1)} \log k - c_k^{-1 - 2\alpha(l)}} \leq e^{-k_{\alpha(l)}}. \quad \hspace{1cm} (4.14)$$
By (4.12) and (4.14) we have
\[ d_{j,k}(\lambda) \leq e^{-k^{\alpha(1)}} \leq e^{-k^{\frac{1}{2}}}, \]
for all \( j, k \in \mathbb{N} \) with \( k \geq g(j) \). It follows by Lemma 6 that the series
\[ \sum_{k=0}^{\infty} d_{j,k}(\lambda) \quad (j = 0, 1, 2, \ldots) \]
are uniformly \( p \)-convergent, i.e., Claim 3 holds.

By (4.1) and Claim 2 we have
\[ X \subseteq \bigcup_{j=0}^{\infty} \bigcap_{t=0}^{\infty} X_{j,k} \subseteq \bigcup_{j=0}^{\infty} \bigcap_{t=0}^{\infty} \bigcup_{k=t}^{\infty} S_{d_{j,k}}. \]

By Claim 1, Claim 3, and Lemma 7, it follows that \( \mu_{\text{pspace}}(X) = 0 \). This completes the proof of Theorem 3 (and the Main Theorem).

\[ \square \]

5 Conclusion

We have used pseudorandom oracles to give a new characterization of BPP. If we write RAND(pspace) for the set of all pspace-random languages, then our characterization implies that \( L \in \text{P}^A \) for every \( L \in \text{BPP} \) and every \( A \in \text{RAND(pspace)} \). This result strengthens the intuition that pspace-random languages are “adequate sources” for all BPP problems. (Earlier, more asymptotic, evidence for this view appears in [17].)

Our work also gives a more detailed analysis of the Bennett and Gill [2] result that \( \text{P}^A = \text{BPP}^A \) for almost every oracle \( A \). Specifically, under every pspace-random oracle \( A \), \( \text{E}^A \) contains languages that are very hard to approximate with oracle circuits. Such a hard language can, by the work of Nisan and Wigderson [21, 22], be used to construct a pseudo-random generator that is quick enough and secure enough to establish \( \text{P}^A = \text{BPP}^A \). Since almost every oracle \( A \) is pspace-random, the result of Bennett and Gill [2] follows.

Acknowledgment

I thank Ron Book and other participants in an informal workshop in structural complexity theory, held in 1990 at the University of California in Santa Barbara, for their helpful remarks on this work. I also thank David Juedes and Elvira Mayordomo for useful observations on resource-bounded measure.

References


