One-Sided Versus Two-Sided Error in Probabilistic Computation

Harry Buhrman[1*] and Lance Fortnow[2**]

[1] CWI, PO Box 94079, 1090 GB Amsterdam, The Netherlands,
buhrman@cwi.nl,
http://www.cwi.nl/~buhrman

[2] University of Chicago, Department of Computer Science,
1100 E. 58th St., Chicago, IL 60637
fortnow@cs.uchicago.edu,
http://www.cs.uchicago.edu/~fortnow

Abstract. We demonstrate how to use Lautemann's proof that **BPP** is in Σ_2^p to exhibit that **BPP** is in $\mathbf{RP}^{\mathbf{PromiseRP}}$. Immediate consequences show that if **PromiseRP** is easy or if there exist quick hitting set generators then $\mathbf{P} = \mathbf{BPP}$. Our proof vastly simplifies the proofs of the later result due to Andreev, Clementi and Rolim and Andreev, Clementi, Rolim and Trevisan.

Clementi, Rolim and Trevisan question whether the promise is necessary for the above results, i.e., whether $\mathbf{BPP} \subseteq \mathbf{RP}^{\mathbf{RP}}$ for instance. We give a relativized world where $\mathbf{P} = \mathbf{RP} \neq \mathbf{BPP}$ and thus the promise is indeed needed.

1 Introduction

Andreev, Clementi and Rolim [ACR98] show how given access to a quick hitting set generator, one can approximate the size of easily describable sets. As an immediate consequence one gets that if quick hitting set generators exist then $\mathbf{P} = \mathbf{BPP}$. Andreev, Clementi, Rolim and Trevisan [ACRT97] simplify the proof and apply the result to simulating **BPP** with weak random sources.

Much earlier, Lautemann [Lau83] gave a proof that $\mathbf{BPP} \subseteq \Sigma_2^p = \mathbf{NP}^{\mathbf{NP}}$, simplifying work of Gács and Sipser [Sip83]. Lautemann's proof uses two simple applications of the probabilistic method to get the existence results needed. As often with the case of the probabilistic method, the proof actually shows that the overwhelming number of possibilities fulfill the needed requirements. With this observation, we show that Lautemann's proof puts **BPP** in the class $\mathbf{RP}^{\mathbf{PromiseRP}[1]}$. Since quick hitting set generators derandomize **PromiseRP** problems, we get the existence of quick hitting set generators implies $\mathbf{P} = \mathbf{BPP}$. This greatly simplifies the proofs of Andreev, Clementi and Rolim [ACR98] and Andreev, Clementi, Rolim and Trevisan [ACRT97].

* Partially supported by the European Union through NeuroCOLT ESPRIT Working Group Nr. 8556, and HC&M grant nr. ERB4050PL93-0516.

** Supported in part by NSF grant CCR 92-53582.

The difference between **RP** and **PromiseRP** is subtle but important. In the class **RP** we require the probabilistic Turing machine to either reject always or accept with probability at least one-half for all inputs. In **PromiseRP** we only need to solve instances where the machine rejects always or accepts with probability at least one-half.

A survey paper by Clementi, Rolim and Trevisan [CRT98] asks whether we can remove the promise in our result, i.e., whether $\mathbf{BPP} \subseteq \mathbf{RP}^{\mathbf{RP}}$. We give a relativized counterexample to this conjecture by exhibiting an oracle A such that $\mathbf{P}^A = \mathbf{RP}^A$ but $\mathbf{P}^A \neq \mathbf{BPP}^A$. Since virtually all the techniques used in derandomization relativize, this means that new techniques will be required to collapse **BPP** in this way.

2 Definitions

We assume the reader familiar with the standard notions of Turing machines, and deterministic, nondeterministic and probabilistic polynomial-time computation. We let Σ represent the binary alphabet $\{0,1\}$.

A quick hitting set generator finds strings in large easily describable sets.

Definition 1. *A* quick δ-hitting set generator *is a polynomial-time computable function h mapping 1^n to a set of strings of length n such that for all n if $f : \Sigma^n \rightarrow \{0,1\}$ is a function computed by circuits of at most n gates and $\Pr_{x\in\Sigma^n}(f(x) = 1) \geq \delta$ then $f(x) = 1$ for some x in $h(1^n)$.*

Andreev, Clementi and Rolim [ACR98] show that for any $\delta, \delta' > 0$, if quick δ-hitting set generators exist than so do δ'-hitting set generators. We will drop δ in this case.

We have many variations of probabilistic complexity classes. In this paper, we will concern ourselves with **RP**, **BPP**, **PromiseRP** and **PromiseBPP**.

Definition 2. *A language L is in the class* **RP** *if there exists a probabilistic polynomial-time Turing machine such that for all $x \in \Sigma^*$,*

- *If x is in L then* $\Pr(M$ *accepts* $x) \geq 1/2$, *and*
- *If x is not in L then* $\Pr(M$ *accepts* $x) = 0$.

Sometimes the class **RP** is denoted simply by **R**.

Definition 3. *A language L is in the class* **BPP** *if there exists a probabilistic polynomial-time Turing machine such that for all $x \in \Sigma^*$,*

- *If x is in L then* $\Pr(M$ *accepts* $x) \geq 2/3$, *and*
- *If x is not in L then* $\Pr(M$ *accepts* $x) \leq 1/3$.

Languages in **RP** require machines M that fulfill the requirements of Definition 2 for all inputs. Sometimes we would like to consider probabilistic machines restricted to inputs where the desired requirements hold. We use **PromiseRP** to describe these problems. This does not form a class per se, but we can formally define the notions of **PromiseRP** being easy and oracle access to **PromiseRP**.

Definition 4. *We say that a language A is* **RP***-consistent with a probabilistic polynomial-time Turing machine M if for all $x \in \Sigma^*$,*

- *x is in A if* $\Pr(M$ *accepts* $x) \geq 1/2$, *and*
- *x is not in A if* $\Pr(M$ *accepts* $x) = 0$.

Note that A may be arbitrary for x such that $0 < \Pr(M \text{ accepts } x) < 1/2$.

Definition 5. *We say* **PromiseRP** *is* easy *if for every probabilistic polynomial-time Turing machine M there is a set A in* **P** *that is* **RP***-consistent with M.*

Using repetition we can reduce the error in Definitions 2-5 to $2^{-q(|x|)}$ for any polynomial q.

Contrast Definition 5 to Definition 2. In particular we have **PromiseRP** is easy implies **P** = **RP**. The converse is not so simply provable, relativized counterexamples easily follow from known results on generic oracles [IN88]. The oracle we develop in Section 4 also gives a relativizable counterexample.

Definition 6. *For any relativizable complexity class $\mathcal{C}$, L is in $\mathcal{C}^{\mathbf{PromiseRP}}$ if there is a probabilistic polynomial-time Turing machine M such that L is in $\mathcal{C}^A$ for all A* **RP***-consistent with M.*

We can also define $\mathcal{C}^{\mathbf{PromiseRP}[k]}$ if we allow only k queries to A in Definition 6. We can use the notation **PromiseBPP** in a similar manner.

One might want to require in Definition 6 that L be in $\mathcal{C}^A$ via a fixed machine depending only on M. Grollmann and Selman [GS88] show that this restriction does not affect Definition 6. For completeness we give a proof of the equivalence of the two definitions in Section 5.

It is not hard to see that there is an easy connection between hitting set generators and **PromiseRP**.

Fact 1 *If there are quick hitting set generator then* **PromiseRP** *is easy.*

3 One-Sided Promise Gives BPP

Theorem 1.

$$\mathbf{BPP} \subseteq \mathbf{RP}^{\mathbf{PromiseRP}[1]}$$

Proof: We basically use the proof of Lautemann [Lau83] that **BPP** is in Σ_2^p to prove Theorem 1.

Let L be a language in **BPP** and M a probabilistic polynomial-time Turing machine accepting L with an error of 2^{-n} on inputs of length n. Let $q(n)$ be the maximum number of coin tosses on any computation path of M on any input of length n. Note $q(n)$ is bounded by a polynomial in n.

Let A be the set of pairs $\langle x, r\rangle$ such that $|r| = q(|x|)$ and $M(x)$ using r as its random coins will accept. Note that A is computable in deterministic polynomial time. We now define the set B as:

$$B = \{\langle x, z_1, \ldots, z_{q(|x|)}\rangle \mid |z_1| = \cdots = |z_{q(|x|)}| = q(|x|) \text{ implies there is some } w \in \Sigma^{q(|x|)} \text{ such that } \langle x, w \oplus z_1\rangle \notin A \wedge \cdots \wedge \langle x, w \oplus z_{q(|x|)}\rangle \notin A\}.$$

Here $u \oplus v$ for $|u| = |v|$ is the bitwise parity of u and v.

Note we have $B \in \mathbf{NP}$. First we will show that L is in $\mathbf{RP}^{B[1]}$. Our $\mathbf{RP}^B$ algorithm on input x with $n = |x|$ simply chooses $z_1, \ldots, z_{q(n)}$ independently at random from $\Sigma^{q(n)}$ and then accepts if $\langle x, z_1, \ldots, z_{q(n)} \rangle$ is not in B.

If x is in L then consider a fixed w and i, $1 \le i \le q(n)$. The probability that $\langle x, w \oplus z_i \rangle$ is not in A is at most 2^{-n}. Since the z_i's are chosen independently, the chance that $\langle x, w \oplus z_i \rangle$ is not in A for every z_i, $1 \le i \le q(n)$ is at most $2^{-nq(n)}$. Since there are $2^{q(n)}$ possible w's we have

$$\Pr(\langle x, z_1, \ldots, z_{q(n)} \rangle \in B) \le 2^{-n}.$$

Now suppose that x is not in L. Fix $z_1, \ldots, z_{q(n)}$ and i, $1 \le i \le q(n)$. If we choose w at random, the probability that $w \oplus z_i$ is in A is at most 2^{-n}. The probability that $w \oplus z_i$ is in A for some i is at most $q(n)2^{-n}$ which for sufficiently large n is much smaller than $1/2$. Thus for every $z_1, \ldots, z_{q(n)}$ of strings of length $q(n)$, $\langle x, z_1, \ldots, z_{q(|x|)} \rangle$ is in B.

Now we wish to show that L is in $\mathbf{RP}^{\mathbf{PromiseRP}[1]}$. Let C be any set such that C and B agree on tuples where the w is chosen at random and the acceptance probability is either zero or greater than one-half.

More specifically $\langle x, z_1, \ldots, z_{q(|x|)} \rangle$ is in C if

1. $|z_i| = q(|x|)$ for each i, $1 \le i \le q(|x|)$, and
2. the number of w of length $q(|x|)$ such that

$$\langle x, w \oplus z_1 \rangle \notin A \wedge \cdots \wedge \langle x, w \oplus z_{q(|x|)} \rangle \notin A$$

   is greater than $2^{q(|x|)-1}$.

The tuple $\langle x, z_1, \ldots, z_{q(|x|)} \rangle$ is not in C if

1. $|z_i| = q(|x|)$ for each i, $1 \le i \le q(|x|)$, and
2. there are no w of length $q(|x|)$ such that

$$\langle x, w \oplus z_1 \rangle \notin A \wedge \cdots \wedge \langle x, w \oplus z_{q(|x|)} \rangle \notin A.$$

The set C can be arbitrary for all other inputs.

The proof above that L is in $\mathbf{RP}^{B[1]}$ also shows that L is in $\mathbf{RP}^{C[1]}$. □

In the proof of Theorem 1, if x is in L and the z_i are badly chosen then the number of w such that

$$\langle x, w \oplus z_1 \rangle \notin A \wedge \cdots \wedge \langle x, w \oplus z_{q(|x|)} \rangle \notin A$$

might be nonzero yet small. This is why we need **PromiseRP** instead of just **RP** for this proof. Theorem 3 shows that any relativizable proof would need to use **PromiseRP**.

From Theorem 1 and its proof we get the following two corollaries.

Corollary 1. *If* **PromiseRP** *is easy then* $\mathbf{P} = \mathbf{BPP}$ *and* **PromiseBPP** *is easy.*

Corollary 2 (Andreev-Clementi-Rolim). *If quick hitting set generators exist then* $\mathbf{P} = \mathbf{BPP}$.

The proof of Theorem 1 only uses the set A restricted to the inputs of the form $\langle x, r \rangle$. Thus we can use **PromiseBPP** is easy instead of just $\mathbf{P} = \mathbf{BPP}$ in Theorem 1 and Corollaries 1 and 2.

Andreev, Clementi and Rolim [ACR98] prove the following stronger result to get Corollary 2.

Theorem 2 (Andreev-Clementi-Rolim). *For any* $\epsilon > 0$, *there is a polynomial-time algorithm that, given access to a quick hitting set generator, and given as input a circuit* C *returns a value* D *such that*

$$|\Pr_{x \in \Sigma^n}(C(x) = 1) - D| \leq \epsilon.$$

We should note that Theorem 2 also follows from Theorem 1. One just need notice that distinguishing the possibilities that $\Pr_{x \in \Sigma^n}(C(x) = 1) \geq D + \epsilon$ and $\Pr_{x \in \Sigma^n}(C(x) = 1) \leq D - \epsilon$ is a **PromiseBPP** question.

4 RP Can Be Easy without BPP Being Easy

In this section we show that Theorem 1 cannot be improved to show that $\mathbf{P} = \mathbf{R}$ implies $\mathbf{P} = \mathbf{BPP}$ using relativizing techniques.

Theorem 3. *There exists a relativized world where* $\mathbf{P} = \mathbf{RP} \neq \mathbf{BPP}$.

Define the following function $tower(0) = 2$, $tower(n + 1) = 2^{tower(n)}$, i.e. $tower(n)$ is an exponential tower of $n + 1$ 2's. We will use a special type of generic (see [FFKL93] for an overview) to prove the theorem.

Definition 7. *A* **BPP**-*generic oracle* G *is a type of generic oracle that is only defined at length* n *such that* $n = tower(m)$ *for some* m. *Moreover at these lengths it will always be the case that at most* $1/3$ *or more than* $2/3$ *of the strings of length* n *are in* G. *We will call oracles that satisfy these requirements oracles that are* **BPP**-*promise.*

The oracle that fulfills the conditions of Theorem 3 will be $\mathbf{QBF} \oplus G$ for G a **BPP**-generic. Here **QBF** is the **PSPACE**-complete set of true quantified boolean formulae. The following lemma shows that the second part of Theorem 3 is fulfilled.

Lemma 1. *Let* G *be a* **BPP**-*generic.* $\mathbf{P}^{\mathbf{QBF} \oplus G} \neq \mathbf{BPP}^{\mathbf{QBF} \oplus G}$.

Proof: This follows because G is generic and the condition that $\mathbf{P} \neq \mathbf{BPP}$ can be met under the **BPP** promise of G. □

The more difficult part is to show that $\mathbf{P}^{\mathbf{QBF} \oplus G} = \mathbf{R}^{\mathbf{QBF} \oplus G}$. We will need the following notion of categoricity.

Definition 8. *A polynomial time nondeterministic machine M is categorically* **R** *if for all* **BPP***-promise oracles B it is the case that for all x $M^{\mathbf{QBF}\oplus B}(x)$ has either more than $1/2$ of its paths accepting or none. We will also call these machines categorical.*

The idea is to show that if M is categorical then there is a polynomial time (relative to **QBF**) algorithm that computes for all x whether $M(x)$ accepts or rejects. The core of this proof will be an argument from Nisan [Nis91].

The proof of Theorem 3 follows from Lemmas 2 and 3. Lemma 2 says that if we have a machine $M(x)$ that is categorically **R** and we only consider oracles A such that at most $1/6$ or at least $5/6$ of the strings of length n are in A then $M^{\mathbf{QBF}\oplus A}(x)$ can be decided in polynomial time relative to $\mathbf{QBF}\oplus A$.

Lemma 2. *Fix an input x and let $n = |x|$. Let $M(x)$ be a categorical machine. For any set A that only contains strings of length n with the promise that either at most $1/6$ or at least $5/6$ of the strings of length n are in A, there exists a deterministic strategy that determines $M^{\mathbf{QBF}\oplus A}(x)$, querying only a fixed polynomial number of strings in A. Moreover this strategy can be computed in a fixed polynomial time relative to $\mathbf{QBF}\oplus A$.*

Proof We follow the lines of the proof of Nisan [Nis91]. Suppose M runs in time $p(n)$. Call any B that fulfills the $1/6$, $5/6$ promise $\mathbf{BPP}_2$-promise. Fix A to be any $\mathbf{BPP}_2$-promise oracle.

The deterministic strategy to determine $M^{\mathbf{QBF}\oplus A}(x)$ works as follows.

Let S_1 contain all the oracles B such that $M^{\mathbf{QBF}\oplus B}(x)$ accepts:

$$S_1 = \{B \mid \Pr(M^{\mathbf{QBF}\oplus B}(x) \text{ accepts}) > 0\}$$

Let S_0 contain all the $\mathbf{BPP}_2$-promise oracles such that $M(x)$ rejects:

$$S_0 = \{C \mid C \text{ is } \mathbf{BPP}_2 \text{ -promise and } \Pr(M^{\mathbf{QBF}\oplus C}(x) \text{ accepts}) = 0\}$$

Let B_1 a set in S_1. Fix any accepting path π of $M^{\mathbf{QBF}\oplus B_1}(x)$ with queries $q_1, \ldots, q_p(n)$ on it and let $b_1, \ldots b_{p(n)}$ be such that $B_1(q_i) = b_i$. Next query $q_1, \ldots, q_{p(n)}$ to A and let $a_1, \ldots a_{p(n)}$ be the answers (i.e. $a_i = A(q_i)$). If for all i it holds that $a_i = b_i$ we know that $M^{\mathbf{QBF}\oplus A}(x)$ accepts and we are done. So assume that this is not the case.

At this point we have the following claim:

Claim. For all $C \in S_0$ at least half of the computation paths of $M^{\mathbf{QBF}\oplus C}(x)$ query a string in $Q = q_1, \ldots, q_{p(n)}$.

Proof Suppose this is not true and that there is a $C \in S_0$ such that less than half of the computation paths of $M^{\mathbf{QBF}\oplus C}(x)$ query a string in Q. Consider the oracle C' which is defined as follows. For all $x \notin Q$, $C'(x) = C(x)$ and for $q_i \in Q$, $C'(q_i) = b_i$. (i.e. C' equals C except for the queries in Q where it equals B_1). Since C was $\mathbf{BPP}_2$-promise it follows that C' is **BPP**-promise. Since $M^{\mathbf{QBF}\oplus C'}(x)$ has at least one accepting path π and it is categorical it follows that at least $1/2$ of its paths are accepting. On the other hand since

$M^{\mathbf{QBF}\oplus C}(x)$ has no accepting paths and more than half of the computation paths do not query anything in Q it follows that less than 1/2 of the paths changed and hence that $M^{\mathbf{QBF}\oplus C'}(x)$ still rejects. A contradiction. □

Next adjust S_0 and S_1 such that they only contain oracles that agree with $A(q_1), \ldots, A(q_{p(n)})$ and repeat the above construction. It follows that in each round we learn the answer to a new query that is queried on at least half of the computation paths. Suppose after $2p(n)$ rounds we have not yet encountered a proof that $M^{\mathbf{QBF}\oplus A}(x)$ accepts. Either all the queries on all the paths of $M^{\mathbf{QBF}\oplus A}(x)$ have been queried or the current S_0 is empty. Let E be the set of queries made to A in all the rounds. We will have that $M^{\mathbf{QBF}\oplus A}(x)$ accepts if and only if $M^{\mathbf{QBF}\oplus (A\cap E)}(x)$ has an accepting path.

To choose the set B_1 in each round we need remember the oracle queries previously made to A. It is not hard to see then that this construction can be carried out in **PSPACE** and reducible to **QBF**. □

Let D be the deterministic strategy that comes out of Lemma 2. The next lemma shows that this strategy also works for **BPP**-promise oracles.

Lemma 3. *For any* **BPP**-*promise oracle A. Let D be the strategy as described in Lemma 2. D will compute correctly $M^{\mathbf{QBF}\oplus A}(x)$.*

Proof Suppose that D does not compute $M^{\mathbf{QBF}\oplus A}(x)$ for some **BPP**-promise A. Suppose that A contains at most 1/3 of the strings of length n. The case where A contains more than 2/3 of the strings of length n can be handled similarly.

Suppose D accepted but did not find an accepting path of $M^{\mathbf{QBF}\oplus A}(x)$. This could only have happened if the final S_0 was empty. Let E be a minimal subset of A consistent with D's queries to A such that $M^{\mathbf{QBF}\oplus E}(x)$ rejects. Since S_0 is empty, E must contain at least $\frac{2^n}{6}$ strings. Removing any string y from E not queried by D will cause $M^{\mathbf{QBF}\oplus (E-\{y\})}(x)$ to accept with probability at least one-half. Thus every string in E not queried by D must occur on at least half of the computation paths of $M^{\mathbf{QBF}\oplus E}(x)$ which cannot happen by a simple counting argument.

Thus the only way the strategy can make an error is when D rejects whereas $M^{\mathbf{QBF}\oplus A}(x)$ accepts. Let $Q = q_1, \ldots, q_{2p(n)^2}$ be the queries made by D. and let $R = r_1, \ldots, r_{p(n)}$ be the queries on some accepting path of $M^{\mathbf{QBF}\oplus A}(x)$. Consider the following set A'. For all $q \in Q$ set $A'(q) = A(q)$, and for $r \in R$ set $A'(r) = A(r)$. For all the other strings x set $A'(x) = 0$. It now follows that A' contains at most a polynomial number of strings of length n and is $\mathbf{BPP}_2$-promise. Moreover since $M^{\mathbf{QBF}\oplus A'}(x)$ has an accepting path it follows that $M^{\mathbf{QBF}\oplus A'}(x)$ accepts. But since all the queries made by D will be the same for A and A' it follows that D still rejects contradicting Lemma 2. □

Proof (of Theorem 3) By Lemma 1 it follows that $\mathbf{P}^{\mathbf{QBF}\oplus G} \neq \mathbf{BPP}^{\mathbf{QBF}\oplus G}$. Let M be any categoric machine that runs in time $p(n)$. let x be any string of length l and let m be the biggest m such that $tower(m) \leq p(n)$. Set $n = tower(m)$. Query all the relevant strings in G of length strictly less than n. Since G is only defined at lengths that are a tower of 2's it follows that the previous relevant length is so small that one can query all those strings in polynomial time. Next apply Lemma 3 and use **QBF** to compute $M^{\mathbf{QBF}\oplus A}(x)$. The last possibility is that $M^{\mathbf{QBF}\oplus G}$ happens to be an **R** machine but it is not categoric. This however can not happen since the genericity of G will diagonalize against such non-categoric machines. (See [BI87]) □

Theorem 3 in combination with Theorem 1 gives a relativized world where **PromiseRP** is not easy but $\mathbf{P} = \mathbf{RP}$. This corollary also follows from work of Impagliazzo and Naor [IN88].

Heller [Hel86] exhibits a relativized world where $\mathbf{BPP} = \mathbf{NEXP}$. One might suspect that the techniques of Heller and those used in the proof of Theorem 3 may lead to an oracle A where $\mathbf{P}^A = \mathbf{RP}^A$ and $\mathbf{BPP}^A = \mathbf{NEXP}^A$. We show this cannot happen.

Theorem 4. *In all relativized worlds, if* $\mathbf{P} = \mathbf{RP}$ *and* $\mathbf{NP} \subseteq \mathbf{BPP}$ *then* $\mathbf{P} = \mathbf{BPP}$.

Proof Zachos [Zac88] shows that if $\mathbf{NP} \subseteq \mathbf{BPP}$ then $\mathbf{NP} = \mathbf{RP}$. We then have $\mathbf{P} = \mathbf{NP} = \Sigma_2^p$ and thus $\mathbf{P} = \mathbf{BPP}$. These arguments all relativize. □

5 Relativizing to PromiseRP

Definition 6 may allow the machine that exhibits L in $\mathcal{C}^{\mathbf{PromiseRP}}$ to depend on A instead of just the underlying probabilistic machine. Grollmann and Selman [GS88] give a general result that implies that disallowing this dependence does not change the class $\mathcal{C}^{\mathbf{PromiseRP}}$. For completeness we give a proof of this result.

For simplicity we will show the equivalence for the class $\mathbf{P}^{\mathbf{PromiseRP}}$. The proof works similarly for many other natural classes such as $\mathbf{RP}^{\mathbf{PromiseRP}}$, $\mathbf{NP}^{\mathbf{PromiseRP}}$, $\mathbf{RP}^{\mathbf{PromiseRP}[1]}$, $\mathbf{P}^{\mathbf{PromiseBPP}}$, etc.

Theorem 5 (Grollmann-Selman). *For every language L and the following are equivalent:*

1. *L is in $\mathbf{P}^{\mathbf{PromiseRP}}$, i.e., there exists a probabilistic polynomial-time Turing machine M such that for all A **RP**-consistent with M, there is a polynomial-time oracle Turing machine N such that $L = L(N^A)$.*
2. *There exist a probabilistic polynomial-time Turing machine M and a polynomial-time oracle Turing machine N such that for all A **RP**-consistent with M, $L = L(N^A)$.*

Proof: (2) is more restrictive than (1). We have to show that (1) implies (2). Fix L in $\mathbf{P}^{\mathbf{PromiseRP}}$ and a M that witnesses this.

Let D be the set of x such that $M(x)$ accepts with probability zero or probability at least one-half. Let E be the set of x such that $M(x)$ accepts with probability at least one-half. We have that A is **RP**-consistent with M if and only if $A \cap D = E$.

Let us assume that (2) fails for M, i.e., for every polynomial-time Turing machine N there is an A such that $A \cap D = E$ and $L \neq L(N^A)$. We will create a set B with $B \cap D = E$ such that for all polynomial-time Turing machines N, $L \neq L(N^B)$. This contradicts that fact that M witnesses L in $\mathbf{P}^{\mathbf{PromiseRP}}$.

Let $N_1, N_2, \ldots$ be an enumeration of the polynomial-time oracle Turing machines.

We create B in stages, in each stage we give a partial setting of whether some strings are or are not in B. Let B_0 be the oracle where all strings in E are put in B_0 and all strings in $D - E$ are put out of B_0. Let $m_0 = 0$.

Our goal at stage i will be to guarantee that for any oracle A extending B_i, $L \neq L(N_i^A)$. At the end of stage i we will have all strings of length less than m_i defined in B_i and only the strings in D of length greater than i will be defined.

Stage $i+1$:

Claim. There exists an **RP**-consistent A extending B_i such that $L \neq L(N_i^A)$.

Proof: Suppose not. Create machine N^C that simulates N_i^C except that on oracle queries of length less than m_i, N will answer them according to B_i. Let C be any **RP**-consistent language. Then N^C will simulate N_i^F where

$$F = (B_i \cap \Sigma^{<m_i}) \cup (C \cap \Sigma^{\geq m_i}).$$

Since $C \cap D = E$ we have that F extends B_i. By the assumption that the claim fails we have $L(N^C) = L(N_i^F) = L$. We now have that $L(N^C) = L$ for all **RP**-consistent C contradicting the assumption that (2) fails. □

Fix an **RP**-consistent A and an x such that $x \in L \Leftrightarrow x \notin L(N_i^A)$. Let m_{i+1} be one more than length the longest oracle query made by $N_i^A(x)$ and let B_{i+1} be the extension of B_i where all strings of length less than m_{i+1} are set according to A. □

Acknowledgments

We thank Noam Nisan for bringing the "**PromiseRP** is easy implies **P** = **BPP**" problem to our attention. We thank Luca Trevisan for suggesting the alternate definition for relativized **PromiseRP** (Definition 6) and the possibility that the two definitions are equivalent.

We thank Luca Trevisan, Noam Nisan, Stuart Kurtz, John Rogers and Steve Fenner for many helpful discussions. Dieter van Melkebeek provided many helpful suggestions on earlier drafts. We also thank Ronald de Wolf for the use of his notes.

References

[ACR98] A. Andreev, A. Clement, and J. Rolim. A new derandomization method. *Journal of the ACM*, 45(1):179–213, Januari 1998.

[ACRT97] A. Andreev, A. Clement, J. Rolim, and L. Trevisan. Weak random sources, hittings sets, and BPP simulations. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 264–272. IEEE, New York, 1997.

[BI87] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 118–126. IEEE, New York, 1987.

[CRT98] A. Clementi, J. Rolim, and L. Trevisan. Recent advances towards proving BPP = P. *Bulletin of the European Association for Theoretical Computer Science*, 64:96–103, February 1998.

[FFKL93] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 120–131. IEEE, New York, 1993.

[GS88] J. Grollmann and A Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17:309–355, 1988.

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Computation*, 71:231–243, 1986.

[IN88] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of the 3rd IEEE Structure in Complexity Theory Conference*, pages 29–38. IEEE, New York, 1988.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

[Nis91] N. Nisan. CREW PRAMSs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, December 1991.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on the Theory of Computing*, pages 330–335. ACM, New York, 1983.

[Zac88] S. Zachos. Probabilistic quantifiers and games. *Journal of Computer and System Sciences*, 36:433–451, 1988.