Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 68 (2001)

P(promiseBPP) = P(APP)

Philippe Moser*

Abstract

We show that for deterministic polynomial time computations, oracle access to **APP**, the class of real functions approximable by probabilistic Turing machines, is the same as having oracle access to promise-**BPP**. First, we construct a mapping that maps every function in **APP** to a promise problem in **prBPP**, and that preserves completeness, i.e. maps complete functions to complete promise problems. Next, we construct a mapping from **prBPP** into **APP**, that maps every promise problem to a function in **APP**, and which preserves completeness. Then, we prove that $\mathbf{P}^{\mathbf{APP}} = \mathbf{P}^{\mathbf{prBPP}}$. Finally, we use our results to simplify proofs of important results on **APP**, such as the **APP**-completeness of the function f_{CAPP}, which approximates the acceptance probability of a Boolean circuit, the error probability reduction Theorem for **APP** functions, and the conditional derandomization result **APP** = **AP** iff **prBPP** is easy.

1 Introduction

The complexity class **BPP** is sometimes considered to be the class of all feasibly computable problems. One reason for this is because randomized algorithms work so well in practice. Though, **BPP** is hard to study as a class because of its semantic nature: on every input, a **BPP** machine must have either at least 3/4 or at most 1/4 accepting paths. Another difficulty is the lack of known complete sets for **BPP**, indeed it has been conjectured that **BPP** does not have any complete sets. One reason for this is the existence of a relativized world where **BPP** (and other semantic classes) do not have complete sets (see [Sip82] and [HH86]).

One way around this difficulty is to consider promise problems, i.e. problems that need to be solved only on instances where a certain promise holds. Thus the canonical complete language $L = \{(M, x, 1^t) |\ M$ is a **BPP** machine and M accepts x in at most t steps$\}$, together with the promise that M is a **BPP** machine, is promise-**BPP** (denoted **prBPP**) complete. Indeed once it is known know that M is a **BPP** machine, a probabilistic algorithm can simulate machine M on input x, thus deciding, with high probability, whether M accepts x or not.

Another approach was introduced by V. Kabanets et al. in [KRC00]. They introduced a natural generalization of **BPP**, namely the class **APP** of real-valued functions $f : \{0,1\}^* \to [0,1]$, that can be approximated within any $\epsilon > 0$, by a probabilistic Turing machine running in time polynomial in the input size and the precision $1/\epsilon$. They showed that **BPP** is exactly the subset of all Boolean functions in **APP**. Moreover they proved that computing the acceptance probability of a given Boolean circuit is an **APP**-complete problem.

*Address: Theorical Computer Science Department, University of Geneva. Email: moser@cui.unige.ch;

ISSN 1433-8092

This paper shows that deterministic polynomial time algorithms, having oracle access to **APP**, are as powerful as those having oracle access to **prBPP**; i.e. $\mathbf{P}^{\mathbf{APP}} = \mathbf{P}^{\mathbf{prBPP}}$. In order to build a mapping between **APP** and **prBPP** we associate each function to its graph. Recall that the graph of a real valued function $f : \{0,1\}^* \to [0,1]$, is defined as being the set of encoded triples $(1^k, x, y)$ such that $f(x) = y$ within distance $1/k$. Our first result states that computing the graph of the **APP**-complete function f_{CAPP} (where f_{CAPP}, on input a Boolean circuit, outputs its probability of acceptance), together with the promise that all queries $f(x) \stackrel{?}{=} y$ made to graph(f_{CAPP}) have the property that the distance between $f(x)$ and y is either "very small", or "rather large", is **prBPP** complete. Then we prove that computing the graph of any function in **APP**, together with the same promise, is in **prBPP**. This yields a mapping from **APP** to **prBPP**, mapping each function in **APP** to a promise problem in **prBPP**, and preserving completeness, i.e. mapping complete functions to complete promise problems.

For the other direction, we first prove that for any real-valued function $f : \{0,1\}^* \to [0,1]$, such that the problems of computing its graph (together with the same promise as above) is in **prBPP**, f is in **APP**. Second we construct a mapping from **prBPP** to **APP**, that maps every promise problem to a real-valued function, and which preserves completeness. Next, we prove that $\mathbf{P}^{\mathbf{APP}} = \mathbf{P}^{\mathbf{prBPP}}$.

Finally we use our results to simplify proofs of important results about **APP**. Namely it is shown in [KRC00] that similarly to the case of **BPP**, the error probability for **APP** functions can be reduced exponentially. Their proof is rather technical and relies on a rather involved argument of repeated trials; on the other hand the idea of our proof is very simple: let f be any function in **APP**. We first map f to its corresponding promise problem (Q, R) in **prBPP**. Then using standard error probability reduction for languages in **prBPP**, we obtain a Turing machine solving (Q, R) with exponentially small error probability. Finally by mapping (Q, R) to its corresponding function in **APP**, we obtain a function f' which is equal to f, and such that there is a Turing machine computing f' with exponentially small error probability.

Second it is proved in [KRC00] that the function f_{CAPP} (where f_{CAPP} on input a Boolean circuit C outputs its probability of acceptance) is **APP**-complete under approximate polynomial time many-one reduction (the many-one reduction version for **APP**). We cannot prove this directly from our results. Still we can prove a slightly weaker result, namely the completeness of f_{CAPP} under polynomial time Turing approximate reduction (the polynomial Turing reduction version for **APP**).

Finally it is proved in [For01] that **APP** = **AP** iff **prBPP** is easy. We prove this by using our two mappings between **APP** and **prBPP**.

2 Preliminaries

Since we are working with real-valued functions, we need the following definition of approximate equality. Let $a, b, \epsilon \in [0,1]$ be real numbers. We say that a and b are ϵ-equal (denoted $a \stackrel{\epsilon}{=} b$) if $d(a,b) := |a - b| \leq \epsilon$.

As usual, a function $f : \{0,1\}^* \to [0,1]$, mapping strings to real numbers, is defined as a family of functions $f = \{f_n\}_{n \geq 0} : \{0,1\}^* \to [0,1]$, where $f_n : \{0,1\}^n \to [0,1]$. V. Kabanets and al. in [KRC00], introduced the class **APP** of real valued function. Here is their definition.

Definition 1 *A family* $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \to [0,1]$ *of real-valued functions is in* **APP**, *if there exists a probabilistic, polynomial-time Turing machine* M, *and a polynomial* $q(k,n)$ *such that,* $\forall k, n \in \mathbb{N}, \forall x \in \{0,1\}^n$,

$$\Pr_{w\in\{0,1\}^{q(k,n)}}[M_w(1^k,x) \stackrel{\frac{1}{k}}{=} f_n(x)] \geq \frac{3}{4}.$$

To simplify notations we will denote $\Pr_{w\in\{0,1\}^{q(k,n)}}$, by $\Pr_w$, where it is implicit that w is a random string of size polynomial in k and n.

Consider the following family of functions $f_{\text{CAPP}} : \{0,1\}^* \to [0,1]$, which takes as an input a Boolean circuit C, and outputs its acceptance probability, i.e. $f_{\text{CAPP}}(C) = \Pr_w[C(w) = 1]$. It was proved in [KRC00] that the function f_{CAPP} is **APP**-complete under polynomial many-one approximate reduction. For two functions f and g in **APP**, f is polynomially many-one approximately reducible to g, denoted $f \lessapprox^{\text{p}}_{\text{mo}} g$, if there is a polynomial family of reductions $r_{n,k} : \{1\}^k \times \{0,1\}^n \to \{0,1\}^{m(k,n)}$, for some polynomial m, such that, $\forall k, n \in \mathbb{N}, \forall x \in \{0,1\}^n$,

$$f_n(x) \stackrel{\frac{1}{k}}{=} g_m(r_{n,k}(1^k, x)).$$

There is also a Turing reduction notion for functions. Let us first give the definition of an oracle for an **APP** function. An oracle for a function $f \in$ **APP** is queried $(1^k, x)$ and answers y, where y is a dyadic rational number of size polynomial in k, and such that $y \stackrel{1/k}{=} f(x)$. For two functions f and g in **APP**, f is polynomially approximately Turing reducible to g, denoted $f \lessapprox^{\text{p}}_{\text{T}} g$, if there is a polynomial time oracle Turing machine M, with oracle access to g, such that, $\forall k, n \in \mathbb{N}, \forall x \in \{0,1\}^n$,

$$f(x) \stackrel{1/k}{=} M^g(1^k, x).$$

The following definitions of promise classes are from Grollmann and Selman [GS88]. A promise problem is a formulation of a partial decision problem that has the structure

Input x Promise $Q(x)$ Property $R(x)$

where Q and R are predicates. Formally, a promise problem is a pair of predicates (Q, R). A Turing machine solves (Q, R) if

$$\forall x[Q(x) \to [M(x) \text{ halts } \wedge [M \text{ accepts } x \leftrightarrow R(x)]]].$$

A solution of (Q, R) is a language A such that,

$$\forall x[Q(x) \to A(x) = R(x)]$$

.

prBPP is the class of all promise problems (Q, R), that have a solution in **BPP** (on instances where the promise is satisfied).

In order to define complete problems for **prBPP** we need to define many-one reductions for promise classes.

Definition 2 *We say that a promise problem (Q, R) is uniformly many-one reducible in polynomial time to a promise problem (S, T), denoted $(Q, R) \leq_{\mathrm{m}}^{\mathrm{p}} (S, T)$, if there exists a partial polynomial time computable function $red : \{x \in \{0,1\}^* \mid Q(x)\} \to \{0,1\}^*$ in* **FP**, *such that for every solution A of (S, T), the set B defined by: $B(x) = A(red(x))$ is a solution of (Q, R).*

Unlike **BPP**, the canonical complete language yields a complete promise problem for **prBPP**. Consider the following promise problem $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}})$.
$\mathcal{Q}_{\mathrm{prBPP}}(M, x, 1^t) = 1$ iff M is a probabilistic Turing machine that decides x **BPP**-wise, i.e. $\mathrm{Pr}_w[M_w(x) = 1] \geq \frac{3}{4}$ or $\leq \frac{1}{4}$.
$\mathcal{L}_{\mathrm{prBPP}}(M, x, 1^t) = 1$ if M accepts x **BPP**-wise in at most t steps, i.e. $\mathrm{Pr}_w[M_w(x) = 1] \geq \frac{3}{4}$.
$\mathcal{L}_{\mathrm{prBPP}}(M, x, 1^t) = 0$ if M rejects x **BPP**-wise in at most t steps, i.e. $\mathrm{Pr}_w[M_w(x) = 1] \leq \frac{1}{4}$.

It is a well-known fact that this promise problem is **prBPP** complete under polynomial time many-one reduction.

Theorem 1 *The promise problem $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}})$ is* **prBPP***-complete under $\leq_{\mathrm{m}}^{\mathrm{p}}$ reduction.*

Proof

i) $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}}) \in$ **prBPP**.

Indeed when $\mathcal{Q}_{\mathrm{prBPP}}(M, x, 1^t)$ holds, we know that machine M has a **BPP** behavior on input x. Therefore a simulation of M on input x yields a **BPP** solution for $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}})$.

ii) $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}})$ is **prBPP**-hard under $\leq_{\mathrm{m}}^{\mathrm{p}}$ reduction.

Let (S, T) be any promise problem in **prBPP**. Let M be a probabilistic polynomial time Turing machine solving (S, T) and let p be its polynomial time bound. Consider the following deterministic polynomial time partial function

$$\begin{cases} red : \{x \in \{0,1\}^* \mid S(x)\} \to \{0,1\} \in \mathbf{FP} \\ x \mapsto (M, x, 1^{p(|x|)}) \end{cases}$$

We claim that red is a many-one reduction from (S, T) to $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}})$. Indeed let A be a solution of $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}})$. It is clear that first if $S(x)$ holds then $\mathcal{Q}_{\mathrm{prBPP}}(M, x, 1^{p(x)})$ holds. Second the set B defined by

$$B(x) = \begin{cases} A(red(x)) & \text{if } \mathcal{Q}_{\mathrm{prBPP}}(x) \\ 0 & \text{otherwise} \end{cases}$$

is a solution of (S, T).

□

3 A mapping from APP to prBPP

Our main tool to build a correspondence between **APP** and **prBPP**, is the graph of a function.

Definition 3 *Let $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \to [0,1]$ be a real valued function. We define its graph by:*

$$gr(f) = \{(1^k, x, y) |\ y \overset{\frac{1}{k}}{=} f(x)\}.$$

Let $f : \{0,1\}^* \to [0,1]$ be a real-valued function. Consider the following promise problem $(\mathcal{P}_f, gr(f))$, where

$$\mathcal{P}_f(1^k, x, y) = \begin{cases} 1 & \text{if } d(f(x), y) \leq \frac{1}{2k} \text{ or } > \frac{3}{2k} \\ 0 & \text{otherwise} \end{cases}$$

For simplicity we will denote $(\mathcal{P}_f, gr(f))$ by $(\mathcal{P}, gr(f))$. The following result states that computing the graph of the function f_{CAPP} (which is **APP**-complete), is a **prBPP**-complete problem.

Theorem 2 *The promise problem $(\mathcal{P}, gr(f_{\text{CAPP}}))$ is* **prBPP***-complete under $\leq_{\text{m}}^{\text{p}}$ reduction.*

Proof
i) $(\mathcal{P}, gr(f_{\text{CAPP}})) \in$ **prBPP**

Let M be a probabilistic transducer witnessing the fact $f_{\text{CAPP}} \in$ **APP**. Consider the following probabilistic polynomial time Turing machine N. On input $(1^k, x, y)$,

- Simulate $M(1^{2k}, x)$ denote the output by $\tilde{y}$.
- Accept iff $d(y, \tilde{y}) \leq \frac{1}{k}$.

It is clear that first N has a **BPP**-like behavior inside the promise. Second it is clear that N decides $gr(f_{\text{CAPP}}))$ correctly inside the promise; indeed by observing Figure 1 we see that wherever y and $\tilde{y}$ are in the interval $[f_{\text{CAPP}}(x) - \frac{1}{2k}, f_{\text{CAPP}}(x) + \frac{1}{2k}]$, N accepts $(1^k, x, y)$ inside the interval $[f_{\text{CAPP}}(x) - \frac{1}{2k}, f_{\text{CAPP}}(x) + \frac{1}{2k}]$, with high probability, and rejects $(1^k, x, y)$ outside the interval $[f_{\text{CAPP}}(x) - \frac{3}{2k}, f_{\text{CAPP}}(x) + \frac{3}{2k}]$, with high probability.

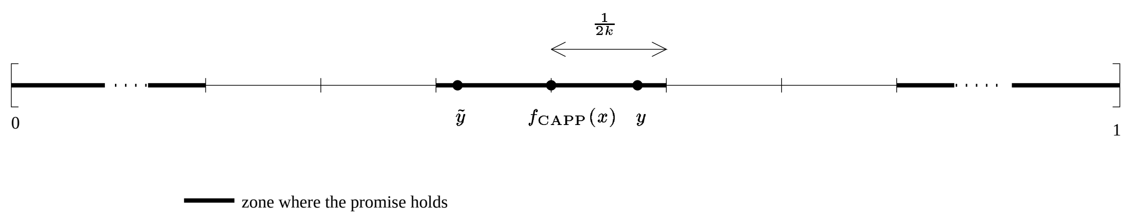

Figure 1: The interval $[0,1]$

ii) $(\mathcal{P}, gr(f_{\text{CAPP}}))$ is **prBPP**-hard.

We prove that $(\mathcal{Q}_{\text{prBPP}}, \mathcal{L}_{\text{prBPP}}) \leq_{\text{m}}^{\text{p}} (\mathcal{P}, gr(f_{\text{CAPP}}))$ which proves the Theorem. Let A be a solution of $(\mathcal{P}, gr(f_{\text{CAPP}}))$. Consider $(N, x, 1^t)$, where N is a probabilistic polynomial time Turing machine such that the promise $\mathcal{Q}_{\text{prBPP}}(N, x, 1^t)$ holds. We construct a solution that computes whether N accepts x. The promise guarantees that N behaves **BPP**-wise on x.

Consider $C_{N,x}$ a Boolean circuit computing $N(x)$, i.e. $C_{N,x}(w) = N_w(x)$. Suppose wlog that N accepts x, i.e. $f_{\mathrm{CAPP}}(C_{N,x}) \in [\frac{3}{4}, 1]$. Consider $k = 2$. Consider the following partial polynomial time computable reduction

$$\begin{cases} red : \{s \in \{0,1\}^* \mid \mathcal{Q}_{\mathrm{prBPP}}(s)\} \to \{0,1\}^* \\ (N, x, 1^t) \mapsto (1^2, C_{N,x}, 1) \end{cases}$$

Consider the set B defined by

$$B(s) = \begin{cases} A(red(s)) & \text{if } \mathcal{Q}_{\mathrm{prBPP}}(s) \\ 0 & \text{otherwise} \end{cases}$$

B is a solution of $(\mathcal{Q}_{\mathrm{prBPP}}, \mathcal{L}_{\mathrm{prBPP}})$. Indeed since $d(f_{\mathrm{CAPP}}(C_{N,x}), 1) \leq \frac{1}{4} \leq \frac{1}{2k}$, the promise for A holds, therefore $(1^2, N(x), 1) \in A$, and B concludes that N accepts x, which is correct.

□

The proof of Theorem 2 can be applied to any function $f \in$ **APP**, thus yielding the following result.

Theorem 3 *Let $f : \{0,1\}^* \to [0,1]$ be a real valued function in* **APP**. *Then $(\mathcal{P}, gr(f))$ is in* **prBPP**.

Proof
Similar to part i) of Theorem 2.

□

4 From prBPP to APP

In a sense Theorem 2 gives a mapping Ψ from **APP** to **prBPP** associating to each real-valued function in **APP** a promise problem in **prBPP**, and preserving completeness (see Theorem 5). The following result gives an inverse for Ψ.

Theorem 4 *Let $f : \{0,1\}^* \to [0,1]$ be a real valued function, such that $(\mathcal{P}, gr(f)) \in$* **prBPP**. *Then f is in* **APP**.

Proof
By hypothesis, there is a solution A which decides $gr(f)$ correctly inside the promise $\mathcal{P}$, moreover $A \in$ **BPP** inside the promise $\mathcal{P}$, i.e. whenever $d(x, f(x)) \leq \frac{1}{2k}$ or $> \frac{3}{2k}$. We construct the following probabilistic polynomial time transducer M for f. On input: $(1^{k'}, x)$,

- Divide the interval $[0,1]$ into $\frac{3k'}{2}$ subintervals of size at most $\frac{2}{3k'}$. Denote by $y_0, \ldots, y_t$ the endpoints.

- Output the first y_i such that $(1^{\frac{3k'}{2}}, x, y_i) \in A$.

We claim that first there is at least one i with $0 \le i \le t$, such that $(1^{\frac{3k'}{2}}, x, y_i) \in A$. Indeed A is correct on input $(1^{\frac{3k'}{2}}, x, y_i)$ when either $d(f(x), y_i) \le \frac{1}{2} \cdot \frac{2}{3k'} = \frac{1}{3k'}$, or, $d(f(x), y_i) > \frac{3}{2} \cdot \frac{2}{3k'} = \frac{1}{k'}$. Moreover we can suppose wlog that $f(x) \in [y_{i_0}, y_{i_0+1}]$; therefore $d(f(x), y_{i_0}) \le \frac{1}{2} \cdot \frac{2}{3k'} = \frac{1}{3k'}$. Therefore $(1^{\frac{3k'}{2}}, x, y_{i_0}) \in A$. Second we prove that when $(1^{\frac{3k'}{2}}, x, y_i) \in A$ then $d(f(x), y_i) \le \frac{1}{k'}$. But this is true because of the promise on A. Indeed if $(1^{\frac{3k'}{2}}, x, y_i) \in A$ then $d(f(x), y_i) \le \frac{3}{2} \cdot (\frac{3k'}{2})^{-1} = \frac{1}{k'}$.

□

We now construct a mapping between **APP** and **prBPP**.

Consider the following two mappings

$$\Psi : \begin{cases} \textbf{APP} \to \textbf{prBPP} \\ f \mapsto (\mathcal{P}, gr(f)) \end{cases} \qquad \Phi : \begin{cases} \textbf{prBPP} \to \textbf{APP} \\ (Q, R) \mapsto f_{Q,R} \end{cases}$$

Where $f_{Q,R}$ is defined as follows. Let $\{M_i\}_{i \ge 1}$ be an enumeration of all probabilistic Turing machines solving (Q, R). Let M' be the first (in lexicographical order). We define $f_{Q,R}(x) = \Pr_w[M'_w(x) = 1]$. The following result states that the two mappings Φ and Ψ preserve completeness.

Theorem 5 *Ψ maps every* **APP** *$\lesssim^{\mathrm{p}}_{\mathrm{m}}$-complete function f to a* **prBPP** *$\le^{\mathrm{p}}_{\mathrm{m}}$-complete problem $(\mathcal{P}, gr(f))$, and Φ maps every* **prBPP** *$\le^{\mathrm{p}}_{\mathrm{m}}$complete problem (Q, R) to an* **APP** *$\lesssim^{\mathrm{p}}_{\mathrm{T}}$-complete function $f_{Q,R}$.*

Proof

For Ψ the result immediately follows from Theorem 2. The Proof for Φ follows.

First we prove that Φ maps $(\mathcal{P}, gr(f_{\mathrm{CAPP}}))$ to a **APP** $\lesssim^{\mathrm{p}}_{\mathrm{T}}$-complete function. Consider $h = \Phi(\mathcal{P}, gr(f_{\mathrm{CAPP}}))$. Let M be the first (in lexicographical order) probabilistic Turing machine solving $(\mathcal{P}, gr(f_{\mathrm{CAPP}}))$. We have $h(1^k, x, y) = \Pr[M_w(1^k, x, y) = 1]$.

Claim: h is **APP** $\lesssim^{\mathrm{p}}_{\mathrm{T}}$-complete.

Proof (of Claim). Let $g \in$ **APP** be any real-valued function, and let N be a probabilistic polynomial Turing machine witnessing this fact. We construct a deterministic polynomial time oracle Turing machine K, such that K^h computes g. Here is a description of K^h on input $(1^k, x)$. Let $red : \{0,1\}^* \to \{0,1\}^*$ be a reduction in **FP** such that $g(x) \overset{\frac{1}{2k}}{=} f_{\mathrm{CAPP}}(red(x))$

- Divide the interval $[0,1]$ into subintervals of size at most $\frac{1}{3k}$. Denote $y_0, y_1, \ldots, y_t$ the endpoints of those subintervals.
- For $i = 0, 1, \ldots, t$, query $h(1^{3k}, red(x), y_i)$ with precision $\frac{1}{10}$. Output the first y_i satisfying

$$h(1^{3k}, red(x), y_i) \ge \frac{3}{4} - \frac{1}{10} \quad (1).$$

Let's prove the correctness of K^h. First we show that there is a y_i satisfying (1). Indeed we can suppose wlog that $f_{\text{CAPP}}(red(x)) \in [y_j, y_{j+1}]$. Therefore wlog $d(f_{\text{CAPP}}(red(x)), y_j) \leq \frac{1}{6k}$. But thanks to the promise, we know that M decides $(1^{3k}, red(x), y_j)$ correctly if $d(f_{\text{CAPP}}(red(x)), y_j) \leq \frac{1}{2} \cdot \frac{1}{3k}$, which is true. Second we show that all y_i satisfying (1) are such that $d(y_i, g(x)) \leq \frac{1}{k}$. Indeed let y_i (where $0 \leq i \leq t$) be any y_i such that $h(1^{3k}, red(x), y_i) \geq \frac{3}{4} - \frac{1}{10}$. Therefore M accepts $(1^{2k}, red(x), y_i)$ which implies, thanks to the promise, that $d(f_{\text{CAPP}}(red(x)), y_i) \leq \frac{3}{2} \cdot \frac{1}{3k} = \frac{1}{2k}$ which implies $d(y_i, g(x)) \leq \frac{1}{k}$.

Second we prove that Φ preserves completeness. So let (S,T) be any **prBPP**-complete language. Therefore let red_2 be a reduction from $(\mathcal{P}, gr(f_{\text{CAPP}}))$ to (S,T). Let N be the first (in lexicographical order) probabilistic polynomial Turing machine that solves (S,T). The following probabilistic polynomial Turing machine M solves $(\mathcal{P}, gr(f_{\text{CAPP}}))$. M on input x computes and outputs $N(red_2(x))$. The end of the proof is similar to the first case.

□

The following result states that deterministic polynomial time algorithms with oracle access to **APP**, are as powerful as those having oracle access to **prBPP**.

Theorem 6 $\mathbf{P}^{\mathbf{APP}} = \mathbf{P}^{\mathbf{prBPP}}$.

Proof

First we prove that $\mathbf{P}^{\mathbf{APP}} \subseteq \mathbf{P}^{\mathbf{prBPP}}$. Let L be any language in $\mathbf{P}^{\mathbf{APP}}$, and let M be a deterministic polynomial time oracle machine, with oracle access to the complete function f_{CAPP}, deciding it. We construct a deterministic polynomial oracle machine N, having oracle access to $(\mathcal{P}, gr(f_{\text{CAPP}}))$ deciding L. On input x, $N^{(\mathcal{P}, gr(f_{\text{CAPP}}))}$ simulates $M^{f_{\text{CAPP}}}(x)$. Suppose that during its computation, $M^{f_{\text{CAPP}}}(x)$ queries string $(1^k, C)$ to its oracle. Then divide the interval $[0,1]$ into subintervals of size at most $\frac{2}{3k}$, denote by $y_0, y_1, \ldots, y_t$ the endpoints of those subintervals. Query whether $(1^{\frac{3k}{2}}, C, y_i) \overset{?}{\in} (\mathcal{P}, gr(f_{\text{CAPP}}))$ for $i = 0, 1, \ldots, t$. Denote by y_j the first y_i such that $(1^{\frac{3k}{2}}, C, y_i) \in (\mathcal{P}, gr(f_{\text{CAPP}}))$. Answer $M^{f_{\text{CAPP}}}$'s query $(1^k, C)$ with y_j.

Second we prove the other inclusion. Let L be any language in $\mathbf{P}^{\mathbf{prBPP}}$, and let M be a deterministic polynomial time oracle machine, with oracle access to the complete set $(\mathcal{P}, gr(f_{\text{CAPP}}))$, deciding it. We construct a deterministic polynomial oracle machine N, with oracle access to f_{CAPP}, deciding L. On input x, $N^{f_{\text{CAPP}}}$ simulates $M^{(\mathcal{P}, gr(f_{\text{CAPP}}))}(x)$. Suppose that during its computation, $M^{(\mathcal{P}, gr(f_{\text{CAPP}}))}(x)$ queries string $(1^k, C, y)$ to its oracle (i.e asking whether $f_{\text{CAPP}}(C) \overset{1/k}{=} y$). Then query $(1^{2k}, C)$ to the oracle for f_{CAPP}, (denote the answer by $\tilde{y}$), and answer $M^{(\mathcal{P}, gr(f_{\text{CAPP}}))}$'s query $(1^k, C, y)$ with "yes" iff $d(\tilde{y}, y) \leq \frac{1}{k}$. It is clear that $N^{f_{\text{CAPP}}}$ answers $M^{(\mathcal{P}, gr(f_{\text{CAPP}}))}$ queries correctly inside the promise $\mathcal{P}$.

□

5 Consequences for APP

Our results significantly simplify the proofs of important results on **APP**. First, it is shown in [KRC00] that similarly to the case of **BPP**, the interval of error probability for functions in

APP can be reduced from $[\frac{1}{2}-p(n), \frac{1}{2}+p(n)]$ to $[2^{-q(n)}, 1-2^{q(n)}]$, for any polynomial $p(n)$ and $q(n)$. We give a much simpler proof using error probability reduction in **prBPP**.

Theorem 7 *Let $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \to [0,1]$ be a family of real-valued functions such that, there exists a probabilistic, polynomial time transducer M and a polynomial p, such that, $\forall k, n \in \mathbb{N}, \forall x \in \{0,1\}^n$,*

$$\Pr_w[M_w(1^k, x) \stackrel{\frac{1}{k}}{=} f_n(x)] \geq \frac{1}{2} + \frac{1}{p(k+n)} \quad ,$$

then for any polynomial q, there exists a probabilistic, polynomial time transducer N, such that $\forall k, n \in \mathbb{N} \forall x \in \{0,1\}^n$,

$$\Pr_{w'}[M_{w'}(1^k, x) \stackrel{\frac{1}{k}}{=} f_n(x)] \geq 1 - 2^{-q(k+n)} \quad .$$

Proof

Let $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \to [0,1]$ be a family of real-valued functions such that, there exists a probabilistic, polynomial time transducer M and a polynomial p, such that, $\forall k, n \in \mathbb{N}, \forall x \in \{0,1\}^n$,

$$\Pr_w[M_w(1^k, x) \stackrel{\frac{1}{k}}{=} f_n(x)] \geq \frac{1}{2} + \frac{1}{p(k+n)} \quad .$$

It is clear that the $[\frac{1}{4}, \frac{3}{4}]$ interval in the definition of the promise problem $(\mathcal{Q}_{\text{prBPP}}, \mathcal{L}_{\text{prBPP}})$ is quite arbitrary and can be replaced by the interval $[\frac{1}{2} - \frac{1}{t(n)}, \frac{1}{2} + \frac{1}{t(n)}]$, where t is any fixed polynomial. Therefore applying Theorem 3, we have that $(\mathcal{P}, gr(f))$ is solved by a probabilistic Turing machine N that accepts (respectively rejects) with probability $\frac{1}{2} + \frac{1}{p(k+n)}$ whenever the promise $\mathcal{P}$ holds. Now let q be any polynomial. Using standard error reduction techniques for **BPP**, we get a probabilistic Turing machine N' such that when the promise $\mathcal{P}$ holds, N' decides the same language as N, and N' accepts (respectively rejects) with probability $\geq 1 - 2^{-q(k+n)}$ (respectively $\leq 2^{-q(k+n)}$). By Theorem 4 we obtain a probabilistic Turing machine M witnessing the fact that $f \in$ **APP** and that errs with probability $2^{-q(k+n)}$.

□

Second it is shown in [KRC00], that the function $f_{\text{CAPP}} : \{0,1\}^* \to [0,1]$ is **APP**-complete, under polynomial time many-one approximate reduction. We cannot prove this directly from our results. Still we can prove a slightly weaker result, namely the completeness of f_{CAPP} under polynomial time Turing approximate reduction.

Theorem 8 *The function f_{CAPP} is* **APP***-complete under polynomial Turing approximate reduction.*

Proof

Let $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \to [0,1]$ be any family of real-valued functions in **APP**. By Theorem 2 we have that $(\mathcal{P}, gr(f)) \in$ **prBPP**. Since the promise problem $(\mathcal{P}, gr(f_{\text{CAPP}}))$ is **prBPP** complete under uniform polynomial many one reduction (Theorem 2), there exists a partial function in **FP**

$$\begin{cases} red : \{s \in \{0,1\}^* | \, \mathcal{P}(s)\} \to \{0,1\}^* \\ (N, x, 1^t) \mapsto (1^{10}, C_{N',x}, 1) \end{cases}$$

such that for any solution A of $(\mathcal{P}, gr(f_{\text{CAPP}}))$, the set

$$B(s) = \begin{cases} A(red(s)) & \text{if } \mathcal{P}(s) \\ 0 & \text{otherwise} \end{cases}$$

is a solution of $(\mathcal{P}, gr(f))$. So let A be a fixed solution of $(\mathcal{P}, gr(f_{\text{CAPP}}))$. We construct a polynomial time deterministic oracle Turing machine N, with oracle access to f_{CAPP}, such that $N^{f_{\text{CAPP}}}$ computes f. Here is a description of $N^{f_{\text{CAPP}}}$ on input $(1^k, x)$.

- Divide the interval $[0,1]$ into $\frac{3k}{2}$ subintervals of size at most $\frac{2}{3k}$. Denote by $y_1, \dots, y_t$ the endpoints of those subintervals.
- Test whether $(1^{\frac{3k}{2}}, x, y_i) \in (P, gr(f))$ for $i = 1, 2, \dots t$, by computing $B(1^{\frac{3k}{2}}, x, y_i) = A(red(1^{\frac{3k}{2}}, x, y_i))$ for $i = 1, 2, \dots t$.
- Output the first y_i such that B accepts $(1^{\frac{3k}{2}}, x, y_i)$.

Let us check that $N^{f_{\text{CAPP}}}$ computes $f(x)$ correctly. So suppose wlog that $f(x) \in [y_i, y_{i+1}]$. Therefore wlog $d(f(x), y_i) \leq \frac{1}{2} \cdot \frac{2}{3k} = \frac{1}{3k}$. Thus there is at least one y_i, where $0 \leq i \leq t$, such that B accepts $(1^{\frac{3k}{2}}, x, y_i)$. Thanks to the promise, we know that B correctly rejects any $(1^{\frac{3k}{2}}, x, z)$ such that $d(f(x), z) > \frac{3}{2} \cdot (\frac{3k}{2})^{-1} = \frac{1}{k}$. Therefore $d(f(x), y_i) \leq \frac{1}{k}$.

□

It is shown in [For01] that **APP** = **AP** iff **prBPP** is easy. We say that **prBPP** is easy if for every promise problem (Q, R) in **prBPP**, there is a language $L \in \mathbf{P}$, such that L decides R when the promise holds, i.e. $[Q(x) \Rightarrow R(x) = L(x)]$.

Theorem 9 **APP** = **AP** *iff* **prBPP** *is easy.*

Proof
Easy consequence of Theorem 6.

□

6 Final remarks

It would be interesting to see whether it is possible, while using our results, to prove the **APP**-completeness of the function f_{CAPP}, under approximate many-one reduction (instead of Turing reduction). The main difficulty here is that even if you are able to compute the graph of the function f_{CAPP}, there is no easy way to compute the image $f_{\text{CAPP}}(x)$, asking only **one** query to its graph.

References

[BDG90] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity II.* EATCS Monographs on Theorical Computer Science Volume 22, Springer Verlag, 1990.

[BDG95] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I.* EATCS Monographs on Theorical Computer Science Volume 11, Springer Verlag, 1995.

[ESY84] S. Even, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61:159–173, 1984.

[For01] L. Fortnow. Comparing notions of full derandomization. *Proceedings of the 16th IEEE Conference on Computational Complexity*, pages 28–34, 2001.

[GS88] J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *Siam Journal on Computing*, 17(2):309–335, April 1988.

[HH86] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Proceedings of the Thirteenth International Colloquium on Automata, Languages, and Programming*, 226 of Lecture Notes in Computer Science:123–135, 1986.

[KF82] K. Ko and H. Friedman. Computational complexity of real functions. *Theorical Computer Science*, pages 20:323–352, 1982.

[Ko91] K. Ko. *Complexity Theory of Real functions.* Birkhäuser, Boston, 1991.

[KRC00] V. Kabanets, C. Rackoff, and S. A. Cook. Efficiently approximable real-valued functions. Technical Report 00-034, Electronic Colloquium on Computational Complexity, April 2000.

[Mos02] P. Moser. Random nondeterministic real functions and Arthur Merlin games. Technical Report 02-006, Electronic Colloquium on Computational Complexity, January 2002.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, New York, 1995.

[Sip82] M. Sipser. On relativization and the existence of complete sets. *Proceedings of the Ninth International Colloquium on Automata, Languages, and Programming*, 140 Lecture Notes in Computer Science:523–531, 1982.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'