Electronic Colloquium on Computational Complexity, Report No. 6 (2002)

Random nondeterministic real functions and Arthur Merlin games

Philippe Moser*

Abstract

We construct a nondeterministic analogue to **APP**, denoted **NAPP**; which is the set of all real valued functions $f : \{0,1\}^* \to [0,1]$, that are approximable within $1/k$, by a probabilistic nondeterministic transducer, in time poly$(n, 1^k)$. We show that the subset of all Boolean functions in **NAPP** is exactly **AM**. We exhibit a natural complete problem for **NAPP**, namely computing the acceptance probability of a nondeterministic Boolean circuit. Then we prove that similarly to **AM**, the error probability for **NAPP** functions can be reduced exponentially. We also give a co-nondeterministic version, denoted **coNAPP**, and prove that all results for **NAPP** also hold for **coNAPP**. Then we construct two mappings between **NAPP** and promise-**AM**, mapping complete problems to complete problems. Finally we show that in the world of deterministic computation, oracle access to **AM** is the same as oracle access to **NAPP**, i.e. $\mathbf{P}^{\mathbf{NAPP}} = \mathbf{P}^{\mathbf{prAM}}$.

1 Introduction

Similarly to the complexity class **BPP**, it is not known whether **AM** (the probabilistic version of **NP**) has complete sets. One reason for this is that **AM** is a semantic class; on every input, there must be at least 3/4 or at most 1/4 random string (of a certain length) that make an **AM** machine accept. Thus the canonical complete language $L = \{(M, x, 1^t)|\ M$ is a **AM** machine and M accepts x in at most t steps$\}$ is not **AM**-complete, because the predicate $-$ *M is a* **AM** *machine* $-$ is undecidable, thus L is not in **AM**.

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 together with the promise that M is indeed an **AM** machine, is promise-**AM** (denoted **prAM**) complete. Indeed once you know that M is an **AM** machine, a probabilistic nondeterministic algorithm can simulate machine M on input x, thus deciding, with high probability, whether M accepts x or not; this puts L in **prAM**.

Another approach was introduced 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**.

In this paper we construct an nondeterministic analogue of **APP** (denoted **NAPP**). We show that similarly to **APP**, the subset of Boolean functions that are in **NAPP**, is exactly

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

ISSN 1433-8092

AM. Then we prove that similarly to **AM**, the error probability for **NAPP** functions can be reduced exponentially, with only a polynomial increase of time. Then we exhibit a natural complete problem for **NAPP**; we show that computing the acceptance of a nondeterministic Boolean circuit is **NAPP**-complete. The crucial point in our definition of **NAPP** is that we say that a nondeterministic probabilistic transducer M computes the image of f on x, iff the largest of all values computed on each nondeterministic branch of M is a good approximation of $f(x)$, with high probability. Thus it is possible to show that such a transducer can approximate, with high probability, the acceptance probability of a nondeterministic circuit C, by simply choosing a random string, and by nondeterministically guessing a witness making C accept. At the end of the computation, only the nondeterministic branches that guessed a correct witness, approximate the acceptance probability of C correctly. Moreover, *all* other nondeterministic branches output values *smaller* than the acceptance probability of C. The same idea applies when proving the error reduction Theorem; by repeated trials, and using Chernoff bounds, we prove that at the end of the computation, the nondeterministic branches that guessed correct witnesses, output a good approximation of $f(x)$ with exponentially small probability error. The other nondeterministic branches output values *smaller* than the values output by the "correct" branches, also with exponentially small probability error.

We also give a co-nondeterministic version of **NAPP**, and show that all results we proved for **NAPP** also hold for its co-nondeterministic analogue.

Then we show that **NAPP** and promise-**AM** are intimately related in the following way. The main tool we use to this purpose is the subgraph of a function. Recall that for a real valued function $f : \{0,1\}^* \to [0,1]$, its subgraph is defined as being the set of triples $(1^k, x, y)$ such that $y \leq f(x)$ within distance $1/k$. Our first result states that computing the subgraph of the **NAPP**-complete function f_{NAPP} (where f_{NAPP} on input a Boolean nondeterministic circuit outputs its probability of acceptance), together with the promise that all queries "$y \leq f(x)$?" made to graph(f_{NAPP}) have the property that the distance between $f(x)$ and y is either "very small" or "rather large", is **prAM** complete. Then we prove that computing the subgraph of any function in **NAPP** together with the same promise, is in **prAM**. This yields a mapping from **NAPP** to **prAM**, mapping each function in **NAPP** to a promise problem in **prAM**, and 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 problem of computing its subgraph (together with the same promise as above) is in **prAM**; f is in **NAPP**. Second we construct a mapping from **prAM** to **NAPP**, that maps every promise problem to a real-valued function, and mapping complete promise problem to complete functions. These mappings can be viewed as a strong connection between random nondeterministic real-valued function, and promise-**AM** on the other hand.

Finally we prove that for deterministic computation, oracle access to **prAM** is the same as oracle access **NAPP**, i.e. $\mathbf{P}^{\mathbf{prAM}} = \mathbf{P}^{\mathbf{NAPP}}$.

2 Preliminaries

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

Definition 1 *A family* $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \rightarrow [0,1]$ *of real-valued functions is in* $\mathbf{N} \cdot \mathbf{AP}$ *(also denoted* $\mathbf{NAP}$ *for nondeterministic-*$\mathbf{AP}$*), if there exists a nondeterministic polynomial-time transducer* M *such that, for all* $k, n \in \mathbb{N}$*, we have*

$$\max_y M(1^k, x, y) \stackrel{\frac{1}{k}}{=} f_n(x),$$

where the max *is taken over all nondeterministic choices* y *of* M*.*

Definition 2 *A family* $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \rightarrow [0,1]$ *of real-valued functions is in* $\mathbf{N} \cdot \mathbf{APP}$ *(also denoted* $\mathbf{NAPP}$*), if there exists a probabilistic, nondeterministic polynomial-time transducer* M *such that, for all* $k, n \in \mathbb{N}$*, we have*

$$\Pr_w[\max_y M_w(1^k, x, y) \stackrel{\frac{1}{k}}{=} f_n(x)] \geq \frac{3}{4},$$

where the max *is taken over all nondeterministic choices* y *of* M*.*

It is not hard to see that **NP** is exactly the subset of all Boolean functions in **NAP**, and **AM** is exactly the subset of all Boolean functions in **NAPP**.

Theorem 1

1. **NP** *is exactly the subset of all Boolean functions in* **NAP***.*

2. **AM** *is exactly the subset of all Boolean functions in* **NAPP***.*

Proof

We prove the second statement. It is easy to see that for a language L in **AM**, its characteristic function χ_L is in **NAPP**. For the other direction, let $f = \{f_n\}_{n\geq 0}$ be a family of Boolean functions in **NAPP**. Then there is a probabilistic, nondeterministic transducer M such that, for all $k, n \in \mathbb{N}$, we have $\Pr_w[\max_y M_w(1^k, x, y) \stackrel{\frac{1}{k}}{=} f_n(x)] \geq \frac{3}{4}$. Fix $k = 3$. So we have: $\Pr_w[\max_y M_w(1^k, x, y) = f_n(x)] \geq \frac{3}{4}$, and since f is Boolean, we have: $\Pr_w[\exists y \; M_w(1^k, x, y) = f_n(x)] \geq \frac{3}{4}$, and therefore $L(M) \in \mathbf{NP}$.

□

To define completeness we need the following definitions of reductions. Let $f, g : \{0,1\}^* \rightarrow [0,1]$ be two functions in **NAPP**. f is polynomial time many-one approximately reducible to g, iff there exists a family of polynomial time computable functions $r_{n,k} : \{1\}^k \times \{0,1\}^n \times \{0,1\}^m$, where m is polynomial in n, k, such that for all $k, n \in \mathbb{N}$,

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

It is easy to check that **NAPP** is closed under polynomial approximate many-one reduction. f is polynomial time Turing reducible to g if there exists a polynomial oracle transducer K, such that for all $k, n \in \mathbb{N}$, and all $x \in \{0,1\}^n$, $f_n(x) \stackrel{1/k}{=} K^g(1^k, x)$; where the oracle for g, on input $(1^k, x)$ outputs a value $1/k$-equal to $g(x)$.

In order to connect functions to languages, we need the subgraph of a real valued function. Let $f = \{f_n\}_{n\geq 0} : \{0,1\}^* \to [0,1]$ be a real valued function. We define its subgraph by: $subgr(f) = \{(1^k, x, y) \in \{1\}^* \times \{0,1\}^* \times \{0,1\}^* |\ \ y \stackrel{1/k}{\leq} f(x)\}$.

Let us recall some definitions about **prAM**, for more details see [ESY84]. Formally, a promise problem is a pair of predicates (Q, R), where Q is the promise, and R is the property. 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 decided by a machine M (i.e. $A = L(M)$) such that M solves (Q, R).

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

In order to define complete problems for **prAM** we need the following definitions of reductions.

Definition 3 *A promise problem (Q, R) is uniformly Turing reducible in polynomial time to a promise problem (S, T), denoted $(Q, R) \leq_{\mathrm{UT}}^{\mathrm{PP}} (S, T)$, if there is a deterministic, polynomial time oracle Turing machine M such that, for every solution A of (S, T), M^A solves (Q, R).*

If machine M depends on the solution A, we simply call it Turing reducibility. Grollmann and Selman [GS88] showed that the two definitions are equivalent. Finally we say that a promise problem (Q, R) is uniformly many-one reducible in polynomial time to a promise problem (S, T), denoted $(Q, R) \leq_{\mathrm{mo}}^{\mathrm{PP}} (S, T)$, if there exists a partial polynomial time computable function $red : \{x \in \{0,1\}^* |\ Q(x)\} \to \{0,1\}^*$ in **FP**, such that for every solution A of (S, T), the set B defined by:

$$B(x) = \begin{cases} A(red(x)) & \text{if } Q(x) \\ \text{undefined} & \text{otherwise} \end{cases}$$

is a solution of (Q, R).

In order to connect **prAM** to **NAPP** we need the following promise problem. Let f be a function in **NAPP**. Consider the following promise problem $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f))$; where $\mathcal{P}_{\mathrm{NAPP}}(1^k, x, y) = 1$ iff $d(x, y) \leq \frac{1}{2k}$ or $> \frac{3}{2k}$; i.e. we promise that all queries to $subgr(f)$ whether $y \leq f(x)$ we make, are such that the distance between y and $f(x)$ is either very small, or rather large.

3 Reduction of the error

Similarly to the case of **AM**, the error probability in Definition 2 can range from $\frac{1}{2} + \frac{1}{p(k+n)}$ to $1 - 2^{-p(k+n)}$, where p is any fixed polynomial.

Theorem 2 *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, nondeterministic transducer M and a polynomial p, such that, $\forall k, n \in \mathbb{N}$,*

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

then for any polynomial q, there exists a probabilistic, nondeterministic transducer N, such that $\forall k, n \in \mathbb{N}$,

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

Proof

Let $x \in \{0,1\}^n$ and $k \in \mathbb{N}$ be fixed. Consider $\epsilon = \frac{1}{3k}$. Consider the following probabilistic, nondeterministic transducer N. On input $(1^k, x)$;

1. Choose $w_1, \ldots, w_m$ at random.

2. Nondeterministically guesses $y_1, \ldots, y_m$.

3. Simulate $M_{w_i}(1^{3k}, x, y_i)$ for $i = 1, \ldots, m$, denote by α_i the output of M on input $(1^{3k}, x, y_i)$, with random seed w_i.

4. Let $\gamma = 2\epsilon$. Let us divide the interval $[0,1]$ in $\frac{1}{\gamma}$ subintervals, of length at most γ. Let $s_0, \ldots, s_l$ be the endpoints of those subintervals. Define

$$s = \begin{cases} \max_{1\leq j\leq l}\{s_j | [s_j - 2\epsilon, s_j + 2\epsilon] \cap [0,1] \text{ contains more than } \frac{m}{2} \text{ of the } \alpha_i\text{'s}\} & (1) \\ 0 \text{ if none such } j \text{ exists} \end{cases}$$

   Finally N outputs s.

For a random string w_i, denote by $y_1^i, \ldots, y_t^i$ the nondeterministic choices for M on input $(1^{3k}, x)$, and by $\alpha_1^i, \ldots, \alpha_t^i$ the outputs of $M((1^{3k}, x, y_j^i)$ for $j = 1, \ldots, t$. Consider the following random variables X_i where $i = 1, \ldots, m$.

$$X_i = 1 \text{ if } \alpha_l^i \stackrel{\epsilon}{\leq} f_n(x) \text{ for } l = 1, \ldots, t \text{ and } \exists l_0 \text{ such that } \alpha_{l_0}^i \stackrel{\epsilon}{=} f_n(x).$$

$X_1, \ldots, X_m$ are independent random variables such that $\Pr[X_i = 1] = p \geq \frac{1}{2} + \frac{1}{p(k+n)}$. Consider $X = \sum_{i=1}^m X_i$. We have $\mu(X) = mp$. Let $\delta = \frac{1}{p(k+n)}$, we have: $(1-\delta)\mu \geq (\frac{1}{2} + \frac{1}{3p(k+n)})$. Choosing $m \geq 4p^2(k+n)q(k+n)$, and using Chernoff bounds, we have $\Pr[X \geq (1-\delta)\mu] \geq 1 - 2^{-q(k+n)}$. We prove that when $X \geq (1-\delta)\mu$, N computes a value $\frac{1}{k}$-equal to $f_n(x)$. Suppose that more than $m(\frac{1}{2} + \frac{1}{3p(k+n)}) > \frac{m}{2}$ of the X_i's are equal to 1, i.e. at the end of a computation path $y_{l_1}^1 \ldots y_{l_m}^m$ of N, a majority of $\alpha_{l_1}^1 \ldots \alpha_{l_m}^m$ are in the interval $I = [f_n(x) - \epsilon; f_n(x) + \epsilon] \cap [0,1]$ (A), or a majority of $\alpha_{l_1}^1 \ldots \alpha_{l_m}^m$ are $\stackrel{\epsilon}{\leq} f_n(x)$ (B).
Case (A): Since there exists a point $\tilde{s} = s_j$ such that $|\tilde{s} - f_n(x)| \leq \frac{\gamma}{2} = \epsilon$, the interval $[\tilde{s} - 2\epsilon; \tilde{s} + 2\epsilon] \cap [0,1]$ contains more than $\frac{m}{2}$ of $\alpha_{l_1}^1 \ldots \alpha_{l_m}^m$ since it contains I as a subinterval. Hence the output of $N(1^k, x, y_{l_1}^1 \ldots y_{l_m}^m)$ satisfies (1). Conversely let $\tilde{s} = s_j$ for a certain $j \in \{1, \ldots, l\}$ be

such that the interval $J = [y - 2\epsilon; y + 2\epsilon] \cap [0,1]$ contains more than $\frac{m}{2}$ of $\alpha^1_{l_1} \dots \alpha^m_{l_m}$. Then J must intersect I, otherwise I would contain less than $\frac{m}{2}$ of $\alpha^1_{l_1} \dots \alpha^m_{l_m}$. Therefore $\tilde{s} \overset{3\epsilon}{=} f_n(x)$, hence $\tilde{s} \overset{\frac{1}{k}}{=} f_n(x)$.

Similarly, for case (B), we have that the output of $N(1^k, x, y^1_{l_1} \dots y^m_{l_m})$ is $\overset{\frac{1}{k}}{\leq} f_n(x)$. Moreover case (A) happens at least once.

4 A complete function

Similarly to the case of **APP**, there is a natural complete function for **NAPP**, under many-one approximate reduction. This is believed not to be true for **AM**, because of its semantic nature. Consider the following family of functions $f_{\mathrm{NAPP}} = \{f_n\}_{n \geq 0} : \{0,1\}^* \to [0,1]$, which takes on input a nondeterministic circuit C, and outputs the following probability: $f_{\mathrm{NAPP}}(C) = \Pr_w[C(w) = 1]$, which is equal to $\Pr_w[\exists y\; C(w,y) = 1]$.

The following Theorem states that the function f_{NAPP} is **NAPP**-complete.

Theorem 3 *The function f_{NAPP} is* **NAPP** *complete, under many-one approximate reduction.*

Proof

The proof is divided in two parts.

Part 1. f_{NAPP} is **NAPP**-hard.

Let $g = \{g_n\}_{n \geq 0} : \{0,1\}^* \to [0,1]$ be any function in **NAPP**, and let M be its transducer, i.e.

$$\Pr_w[\max_y M_w(1^k, x, y) \overset{\frac{1}{k}}{=} g_n(x)] \geq 1 - 2^{-q(k+n)} \quad (1), \text{ which implies}$$
$$\Pr_w[\max_y M_w(1^k, x, y) \overset{\frac{1}{k}}{\leq} g_n(x)] \geq 1 - 2^{-q(k+n)} \quad (2).$$

Consider the following nondeterministic, probabilistic transducer $\tilde{M}$. On input $(1^k, x)$,

1. Choose w at random.
2. Guess y nondeterministically.
3. Compute α_y the output of $M_w(1^{2k}, x, y)$.
4. Output 1 with probability α_y and 0 with probability $1 - \alpha_y$.

By encoding the transducer $\tilde{M}$ into a Boolean circuit, we obtain the following nondeterministic circuit $C = C_{k,x}$: $C(w,y) = \tilde{M}_w(1^k, x, y)$. We have:

$\Pr_w[C(w) = 1] = \Pr_w[\max_y C(w,y) = 1] = \mathrm{E}_w[\max_y M_w(1^{2k}, x, y)] \overset{\frac{1}{k}}{=} g_n(x)$.

Part 2. $f_{\mathrm{NAPP}} \in \mathbf{NAPP}$.

Consider the following probabilistic, nondeterministic transducer M for f_{NAPP}. Input $(1^k, C)$, where C is a nondeterministic circuit.

1. Choose $w_1, \dots, w_m$ at random.
2. Nondeterministically guess $y_1, \dots, y_m$.
3. Compute α_i, where $\alpha_i = C(w_i, y_i)$, for $i = 1, 2, \dots, m$.
4. Output the probability $\frac{1}{m} \sum_{i=0}^{m} \alpha_i$.

Let $p = \Pr_w[\exists y\ C(w, y) = 1]$. Consider the following random variables X_i for $i = 1, 2, \dots, m$.

$$X_i = \begin{cases} 1 & \text{if } w_i \in A \\ 0 & \text{if } w_i \in R \end{cases}$$

Where $A = \{w | \exists y\ C(w, y) = 1\}$ and $R = \{w | \forall y\ C(w, y) = 0\}$.

We have $\Pr[X_i = 1] = p$. By letting $X = \sum_{i=1}^{m} X_i$,we have $\mu = \mathrm{E}(X) = mp$. By using Chernoff bounds, we get:

$$\Pr_{w_1, \dots, w_m}[X < (1 - \tfrac{1}{k})p] \leq 2^{O(\frac{m}{k^2})}, \text{ and}$$
$$\Pr_{w_1, \dots, w_m}[X < (1 - \tfrac{1}{k})p] \leq 2^{O(\frac{m}{k^2})}.$$

We prove that M computes f_{NAPP} correctly when $(1 - \frac{1}{k})p \leq X \leq (1 + \frac{1}{k})p$.

Therefore suppose 4 holds. Without loss of generality, we can suppose $w_1, \dots, w_t$ are in A, and $w_{t+1}, \dots, w_m$ are in R. Let $\bar{y}_1, \dots, \bar{y}_t$ be witnesses for $w_1, \dots, w_t$, and let $\bar{y}_{t+1}, \dots, \bar{y}_m$ be any nondeterministic choices. We call $\bar{y}_1, \dots, \bar{y}_m$ a good path. We show that the value computed on any path of M is $\frac{1}{k}$-smaller than the value computed on a good path. But this is clear: let $y_1, \dots, y_m$ be any path of M, so we have that $C(w_i, \bar{y}_i) \geq C(w_i, y_i)$ for $i = 1, \dots, m$. Moreover, when 4 holds, there exists at least one good path for M. This ends the proof.

□

5 Random co-nondeterministic functions

We now define a co-nondeterministic analogue to **NAPP**, that will satisfy the same relationships with **coNP**, as **NAPP** did with **NP**.

We say that a family $f = \{f_n\}_{n \geq 0} : \{0,1\}^* \to [0,1]$ of real-valued functions is in **coNAPP**, if there exists a probabilistic, nondeterministic polynomial-time transducer M such that, for all $k, n \in \mathbb{N}$, we have

$$\Pr_w[\min_y M_w(1^k, x, y) \stackrel{\frac{1}{k}}{=} f_n(x)] \geq \frac{3}{4},$$

where the min is taken over all nondeterministic choices y of M.

All the results that hold for **NAPP** also hold for **coNAPP**. More precisely:

Theorem 4 **coNP** *is exactly the subset of all Boolean functions in* **coNAPP**.

The proof is similar to that of Theorem 1.

The reduction of the error is also possible and we have the same result as Theorem 2 for **coNAPP**.

For the complete function for **coNAPP**, things change slightly. Consider the following family of functions $f_{\mathbf{coNAPP}} = \{f_n\}_{n\geq 0} : \{0,1\}^* \to [0,1]$, which takes on input a co-nondeterministic circuit C, and outputs the following probability: $f_{\mathbf{coNAPP}}(C) = \Pr_w[C(w) = 1]$, which is equal to $\Pr_w[\forall y\ C(w, y) = 1]$.

Again the function $f_{\mathbf{coNAPP}}$ is **coNAPP**-complete.

Theorem 5 *The function* $f_{\mathbf{coNAPP}}$ *is* **coNAPP**-*complete, under many-one aproximate reduction.*

The proof is similar to that of Theorem 3. The complete functions f_{NAPP} and f_{coNAPP} are connected in the following result.

Lemma 1 *Let C be a nondeterministic circuit, and let C' be the nondeterministic circuit computing $\neg C$. Then,*

$$f_{\mathrm{coNAPP}}(C) = 1 - f_{\mathrm{NAPP}}(C').$$

Proof

$f_{\mathrm{coNAPP}}(C) = \Pr_w[\forall y C(w, y) = 1] = 1 - \Pr_w[\exists y C(w, y) = 0] = 1 - \Pr_w[\exists y \neg C(w, y) = 1] = 1 - f_{\mathrm{NAPP}}(C')$

□

6 A mapping between promise-AM and NAPP

The following result states that the problem of computing the subgraphe of the **NAPP**-complete function f_{NAPP}, together with a promise on the distance between the querries and the value of the function, is **prAM**-complete. This establishes a strong connection between random nondeterministic real functions and **prAM**.

Theorem 6 $(\mathcal{P}_{\mathrm{NAPP}}, \mathit{subgr}(f_{\mathrm{NAPP}}))$ *is* **prAM**-*complete under* $\leq^{\mathrm{PP}}_{\mathrm{mo}}$ *reduction.*

Proof

i) $(\mathcal{P}_{\mathrm{NAPP}}, \mathit{subgr}(f_{\mathrm{NAPP}})) \in$ **prAM**

Consider $f_{\mathrm{NAPP}} \in$ **NAPP** and let M be its probabilistic nondeterministic polynomial transducer. We construct a probabilistic nondeterministic polynomial Turing machine N, solving $(\mathcal{P}_{\mathrm{NAPP}}, \mathit{subgr}(f_{\mathrm{NAPP}}))$. Input $(1^k, x, z)$. (N has to determine whether $z \overset{1/k}{\leq} f_{\mathrm{NAPP}}(x)$).

1. Choose w at random.

2. Nondeterministically guess y.

3. Simulate $M_w(1^{2k}, x, y)$; denote its output by $\tilde{z}$.

4. Accept iff $z \overset{1/k}{\leq} \tilde{z}$.

It is clear that first N has a **AM**-like behaviour inside the promise. Second it is clear that N decides $subgr(f_{\mathrm{NAPP}})$ correctly inside the promise; indeed by observing Figure 1 we see that wherever y and $\tilde{y}$ are in the interval $[f_{\mathrm{NAPP}}(x) - \frac{1}{2k}, f_{\mathrm{NAPP}}(x) + \frac{1}{2k}]$, N always accepts $(1^k, x, y)$ inside the interval $[f_{\mathrm{NAPP}}(x) - \frac{1}{2k}, f_{\mathrm{NAPP}}(x) + \frac{1}{2k}]$, and always rejects $(1^k, x, y)$ outside the interval $[f_{\mathrm{NAPP}}(x) - \frac{3}{2k}, f_{\mathrm{NAPP}}(x) + \frac{3}{2k}]$.

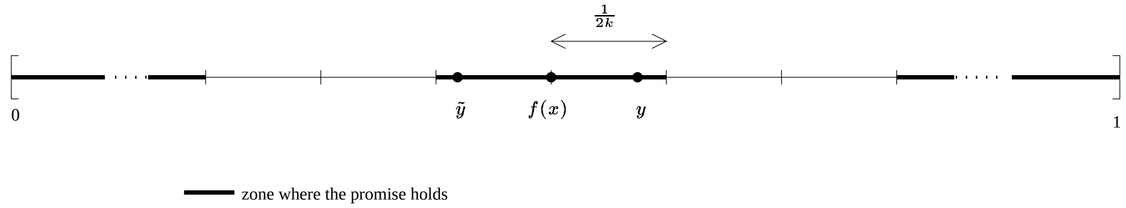

Figure 1: The intervall $[0, 1]$

ii) $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))$ is **prAM**-hard under $\leq_{\mathrm{mo}}^{\mathrm{PP}}$ reduction.

Let $(Q, R) \in$ **prAM** be any promise problem and let M be a probabilistic nondeterministic Turing machine solving it. We construct a polynomial time reduction $r : \{0,1\}^* \to \{0,1\}^*$. Let $x \in \{0,1\}^n$. Let C be the nondeterministic circuit that on input (w, y) computes $M_w(x, y)$. Define $r(x) = (1^{10}, C, 0.5)$. It is easy to see that r is a reduction from (Q, R) to $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))$.

□

The proof of Theorem 6 can be applied to any function $f \in$ **prAM**.

Theorem 7 *Let f be in* **NAPP**. *Then $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f)) \in$* **prAM**.

Theorem 6 gives a mapping Ψ from **NAPP** to **prAM**, associating to each real-valued function in **NAPP** a promise problem in **prAM**, and mapping complete function onto complete promise problems (see Theorem 9). The following result gives an inverse for Ψ.

Theorem 8 *Let $f : \{0,1\}^* \to [0,1]$ be a real valued function, such that $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f)) \in$* **prAM**. *Then f is in* **NAPP**.

Proof

By hypothesis, there is a solution A which decides $subgr(f)$ correctly inside the promise, moreover $A \in$ **prAM** inside the promise, i.e. whenever $d(x, f(x)) \leq \frac{1}{2k}$ or $> \frac{3}{2k}$. Let N be a probabilistic nondeterministic Turing machine deciding A. We construct the following probabilistic nondeterministic polynomial time transducer M for f. 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.
- Nondeterministically find the largest y_i (denoted by y_{i_0}) such that $(1^{\frac{3}{2k}}, x, y_i) \in A$.
- Output y_{i_0}.

Suppose $f(x) \in [y_{i_0}, y_{i_0+1}]$. Wlog $d(f(x), y_{i_0}) \leq \frac{1}{2} \cdot \frac{2}{3k} = \frac{1}{3k}$. Thanks to the promise, A is correct on $(1^{\frac{3}{2k}}, x, y_{i_0})$, and therefore at least one nondeterministic branch of M outputs the value y_{i_0}, with high probability, and $d(y_{i_0}, f(x)) \leq \frac{1}{k}$. Let us prove that with high probability, the bigest value over all nondeterministic branches of M, is $\overset{1/k}{\leq} f(x)$. Thanks to the promise, A's error zone is smaller than $\frac{3}{2} \cdot \frac{2}{3k} = \frac{1}{k}$. Therefore any falsely accepted input $((1^{\frac{3}{2k}}, x, y_i))$ of A, must satisfy $d(f(x), y_i) \leq \frac{1}{k}$. Therefore the largest value over all nondeterministic branches of M, is $\overset{1/k}{\leq} f(x)$, with high probability.

□

We now construct two mappings between **NAPP** and **prAM**.
Consider the following two mappings

$$\Psi : \begin{cases} \mathbf{NAPP} \rightarrow \mathbf{prAM} \\ f \mapsto (\mathcal{P}_{\mathrm{NAPP}}, subgr(f)) \end{cases} \quad \Phi : \begin{cases} \mathbf{prAM} \rightarrow \mathbf{NAPP} \\ (Q, R) \mapsto f_{Q,R} \end{cases}$$

Where $f_{Q,R}$ is defined as follows; Let $\{M_i\}_{\{i \in \mathbb{N}\}}$ be an enumaration of all probabilistic non-deterministic 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 map complete problems to complete problems.

Theorem 9 *Ψ maps every* **NAPP** $\lessapprox^{\mathrm{P}}_{\mathrm{mo}}$*-complete function f to a* **prAM** $\leq^{\mathrm{PP}}_{\mathrm{mo}}$*-complete problem $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f))$, and Φ maps every* **prAM** $\leq^{\mathrm{PP}}_{\mathrm{mo}}$*complete problem (Q, R) to a* **NAPP** $\leq^{\mathrm{P}}_{\mathrm{T}}$*-complete function $f_{Q,R}$.*

Proof
For Ψ the result immediately follows from Theorem 6. The Proof for Φ follows.
First we prove that Φ maps $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))$ to a **NAPP** $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete function. Denote $h = \Phi(\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))$. Let M be the first (in lexicographical order) probabilistic Turing machine solving $(\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))$. We have $h(1^k, x, y) = \Pr[M_w(1^k, x, y) = 1]$.

Claim: h is **NAPP** $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete.

Proof (of Claim). Let $g \in$ **NAPP** 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 $red : \{0,1\}^* \rightarrow \{0,1\}^*$ be a reduction in **FP** such that $g(x) \overset{\frac{1}{2k}}{=} f_{\mathrm{NAPP}}(red(x))$

- Divide the interval $[0,1]$ into subintervals of size at most $\frac{1}{3k}$. Denote $y_0, y_1, \dots, y_t$ the endpoints of those subintervals.
- For $i = 0, 1, \dots, t$ querry $h(1^{3k}, red(x), y_i)$ with precision $\frac{1}{10}$. Output the largest y_i satisfying
$$h(1^{3k}, red(x), y_i) \geq \frac{3}{4} - \frac{1}{10} \quad (1).$$

Let's prove the correctness of K^h. First we show that there is a y_i satsfying (1). Indeed we can suppose wlog that $f_{\mathrm{NAPP}}(red(x)) \in [y_j, y_{j+1}]$. Therefore wlog $d(f_{\mathrm{NAPP}}(red(x)), y_j) \leq \frac{1}{6k}$. But thanks to the promise, we know that M decides $(1^{3k}, red(x), y_j)$ correctly if $d(f_{\mathrm{NAPP}}(red(x)), y_j) \leq \frac{1}{2} \cdot \frac{1}{3k}$, which is true. Second we prove that the largest y_i satisfying (1) is such that $y_i \overset{1/k}{\leq} g(x)$. Thanks to the promise, the error zone is smaller than $\frac{3}{2} \cdot \frac{1}{3k} = \frac{1}{2k}$. Therefore the largest y_i that K^h might output is y_{j+2}, which is still correct, since $d(f_{\mathrm{NAPP}}(red(x)), y_{j+2}) \leq \frac{1}{2k}$, which guarantees $d(g(x), y_{j+2}) \leq \frac{1}{k}$.

Second we prove that Φ maps every complete problem to a complete function. So let (S,T) be any **prAM**-complete language. Therefore let red_2 be a reduction from $(\mathcal{P}_{\mathrm{NAPP}}, gr(f_{\mathrm{NAPP}}))$ 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 $(\mathcal{P}_{\mathrm{NAPP}}, gr(f_{\mathrm{NAPP}}))$. M on input x computes and outputs $N(red_2(x))$. The end of the proof is similar to the first case.

□

7 prAM and NAPP are the same, relative to P

We now prove that for deterministic polynomial computation, oracle access to **prAM** is the same as oracle access to **NAPP**, **coNAPP** or **BP** $\cdot$ **coNP**.

Theorem 10 $\mathbf{P}^{\mathbf{prAM}} = \mathbf{P}^{\mathbf{NAPP}} = \mathbf{P}^{\mathbf{coNAPP}} = \mathbf{P}^{\mathbf{BP}\cdot\mathbf{coNP}}$.

Proof

First we prove the $\supseteq$ inclusion (for the first equality). Let L be any language in $\mathbf{P}^{\mathbf{NAPP}}$, and let $M^{f_{\mathrm{NAPP}}}$ be a deterministic polynomial oracle Turing machine deciding L. We construct a deterministic polynomial oracle machine $N^{(\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))}$ deciding L. On input x, $N^{(\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))}$ simulates $M^{f_{\mathrm{NAPP}}}(x)$. Suppose that during its computation, $M^{f_{\mathrm{NAPP}}}$ querries the string $(1^k, C)$ to its oracle (i.e. asking $f_{\mathrm{NAPP}}(C) \overset{1/k}{=} ?$). Then divide the interval $[0,1]$ into subintervals of size at most $\frac{2}{3k}$. Denote by $y_0, y_1, \dots, y_t$ the endpoints of those subintervals. Querry the oracle whether $(1^{\frac{3k}{2}}, C, y_i) \in (\mathcal{P}_{\mathrm{NAPP}}, subgr(f_{\mathrm{NAPP}}))$, for $i = 0, 1, \dots, t$. Denote by y_{t_0} the largest y_i accepted by the oracle. Answer $M^{f_{\mathrm{NAPP}}}$'s querry $(1^k, C)$ with y_{t_0}. This ends the description of N on input x. Let us prove that N answers M's querries correctly. Suppose $f_{\mathrm{NAPP}}(C) \in [y_j, y_{j+1}]$ for a certain j. Wlog we have $d(y_j, f_{\mathrm{NAPP}}(C)) \leq \frac{1}{3k}$. Therefore y_j will be accepted by N's oracle. Moreover suppose y_{t_0} is the largest y_i accepted by N's oracle. The promise $\mathcal{P}_{\mathrm{NAPP}}$ gurantess that $d(y_{t_0}, f_{\mathrm{NAPP}}(C)) \leq \frac{3}{2} \cdot \frac{2}{3k} = \frac{1}{k}$.

Second we prove the $\subseteq$ inclusion. Let L be any language in $\mathbf{P}^{\mathbf{prAM}}$, and let $M^{\mathbf{prAM}}$ be a deterministic polynomial oracle Turing machine deciding it. We construct a deterministic polynomial oracle machine N^{NAPP} deciding L. On input x $N^{\mathbf{NAPP}}$ simulates $M^{\mathbf{prAM}}(x)$. Suppose $M^{\mathbf{prAM}}$ querries wether $z \stackrel{?}{\in} (Q, R)$ to its oracle, where $(Q, R) \in \mathbf{prAM}$. Since $(Q, R) \in \mathbf{prAM}$, let K be a probabilistic nondeterministic polynomial Turing machine witnessing it. N constructs a nondeterministic circuit C that computes K on input z (i.e. $C(w, y) = K_w(z, y)$). Then N asks its oracle for f_{NAPP}, the value of $f_{\text{NAPP}}(C)$, with precision $\frac{1}{10}$. Finally N answers M's querry z with "yes" iff $f_{\text{NAPP}}(C) \geq \frac{3}{4} - \frac{1}{10}$. This ends the description of N on input x. It is clear that N answers M's querries correctly inside the corresponding promises.

The second equality follows from Lemma 1. The last one is trivial.

$\square$

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*, pages 159–173, 1984.

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

[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, 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.

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

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'