On the Random-Self-Reducibility of Complete Sets

Preliminary Version*

Joan Feigenbaum
AT&T Bell Labs, Rm 2C473
600 Mountain Avenue
Murray Hill, NJ 07974 USA
jf@research.att.com

Lance Fortnow†
University of Chicago
Computer Science Department
Chicago, IL 60637 USA
fortnow@cs.uchicago.edu

Abstract

Informally, a function f is *random-self-reducible* if the evaluation of f at any given instance x can be reduced in polynomial time to the evaluation of f at one or more *random* instances y_i. A set is random-self-reducible if its characteristic function is. Random-self-reducibility plays a major role in many areas of theoretical computer science, including average-case complexity, lower bounds, interactive proof systems, zero-knowledge, instance-hiding, pseudorandom number generation, and program checking.

In this paper, we generalize the previous formal definitions of random-self-reducibility. We show that, even under our very general definition, sets that are complete for any level of the polynomial hierarchy are not random-self-reducible, unless the hierarchy collapses. In particular, NP-complete sets are not random-self-reducible, unless the hierarchy collapses at the third level.

By contrast, we show that sets complete for the classes PP and $\mathrm{MOD}_m\mathrm{P}$ are random-self-reducible.

1 Introduction

Informally, a function f is *random-self-reducible* if the evaluation of f at any given instance x can be reduced in polynomial time to the evaluation of f at one or more *random* instances y_i.

Random-self-reducible functions have many applications, including:

Average-case complexity: A random-self-reduction maps an arbitrary, worst-case instance x in the domain of f to a set of random instances $y_1, \ldots, y_k$ in such a way that $f(x)$ can be computed in polynomial-time, given x, $f(y_1)$, ..., $f(y_k)$, and the coin-toss sequence used in the mapping. Thus the average-case complexity of f, where the average is taken with respect to the induced distribution on instances y_i, is the same, up to polynomial factors, as the worst-case complexity of f. An important special case is that in which each random instance y_i is uniformly distributed over all elements in $Dom(f)$ that have length $|x|$. In this case, f "is as hard on average as it is in the worst case." For example, the PERM (permanent of integer matrices) function is $\#P$-complete (cf. [26]) and has a random-self-reduction in which the induced distribution on each random instance y_i is uniform (cf. [19]). Thus, if the PERM function could be computed efficiently in the average case, *every* function in $\#P$ could be computed efficiently in the *worst* case; furthermore, the random-self-reduction for PERM is very simple, whereas standard average-case hardness proofs are often complicated.

Lower bounds: The random-self-reducibility of the parity function is used in [2] to obtain a simple proof that a random oracle separates the polynomial hierarchy (PH) from PSPACE. (An earlier proof of this result in [12] does not use random-self-reducibility.)

Interactive proof systems and program checkers: Random-self-reductions are crucial

*Final version to appear in *SIAM Journal on Computing*.

†Supported in part by NSF Grant CCR-9009936.

ingredients in many of the original examples of interactive proof systems and program checkers (cf. [9, 17]). Intuitively, this is because the verifier/checker interrogates the prover/program by comparing its output on the specific input of interest to its outputs on other correlated random instances. Several variations of this relationship between random-self-reducibility and proof systems/checkers are stated formally in [10, 20, 25]. These ideas play a crucial role in the characterization of the language-recognition power of interactive proof systems (cf. [4, 20, 22]). Currently, one of the most important open questions about checkability is whether NP-complete sets are checkable. The main result that we present in Section 3 implies that, if NP-complete sets are checkable, their checkers must use radically different techniques from those used by the existing checkers.

Cryptographic protocols: The fact that certain number-theoretic functions are random-self-reducible (and hence hard on average if they are hard at all) is used extensively in the theory of cryptography – e.g., to achieve *probabilistic encryption* (cf. [16]) and *cryptographically strong pseudorandom number generation* (cf. [11]). Random-self-reductions also provide natural examples of *instance-hiding schemes* (cf. [1, 6, 7]), in which a weak, private computing device uses the resources of a powerful, shared computing device without revealing its private data.

Random-self-reductions were defined formally and studied in their own right for the first time by Abadi, Feigenbaum, and Kilian [1]; they considered reductions that map the given instance x to *one* random instance y. It is a corollary of the main result in [1] that no NP-hard function is random-self-reducible in this sense, unless the PH collapses at the third level.

Random-self-reductions that produce several, correlated random instances $y_1, \ldots, y_k$ were defined formally by Feigenbaum, S. Kannan, and Nisan [14]; however, they only considered reductions that produce y_i's that are uniformly distributed over $\{0,1\}^{|x|}$. Their main result is that self-reductions that map x to two instances y_1 and y_2, each of which is uniformly distributed over $\{0,1\}^{|x|}$, do not exist for NP-hard functions, unless the PH collapses at the third level.

The related idea of mapping an instance x in the domain of f to one or more random instances $y_1, \ldots, y_k$ in the domain of a different function g is studied in [1, 6, 7]. For the case of one random y, a negative result for NP-hard functions is obtained in [1]. If multiple random y_i's are allowed, then *every* function f can be *locally randomly reduced* to a related function g; see [6, 7] for a thorough discussion.

In this paper, we continue the study of random-self-reductions from a complexity-theoretic point of view. We further generalize the formal definition of random-self-reducibility that is studied in [14]. Specifically, we look at reductions that map a given instance x to a sequence of random instances $y_1, \ldots, y_k$, with the property that the induced distribution on each y_i depends only on the length of x. We consider both *nonadaptive k-random-self-reductions*, in which the k random instances are produced in one pass, and *adaptive k-random-self-reductions*, in which the instance y_i may depend not only on x and the coin-toss sequence used in the reduction, but on $f(y_1), \ldots, f(y_{i-1})$ as well. Our main results are:

- If S is complete for Σ_i^p, for any $i \geq 1$, and χ_S is nonadaptively $k(n)$-random-self-reducible, for any polynomially bounded function k, then the PH collapses at the $(i+2)^{nd}$ level. In particular, if the characteristic function for any NP-complete set has a nonadaptive random-self-reduction, then the PH collapses at the third level. This strengthens the main result in [14].
- If S is complete for PP or for $\mathrm{MOD}_m\mathrm{P}$, for any $m > 1$, then χ_S is adaptively $k(n)$-random-self-reducible, for some polynomially bounded function k. Setting $m = 2$, we get that $\oplus$P-complete sets are random-self-reducible.

The rest of this paper is organized as follows. In Section 2, we define our terms precisely and recall known results that we will use. Section 3

contains our main negative result about complete sets in the PH. Section 4 contains the proofs that complete sets for PP and $\mathrm{MOD}_m\mathrm{P}$ are random-self-reducible. Open problems are stated in Section 5.

Most of the content of this paper first appeared in our Technical Report [13].

2 Preliminaries

Throughout this paper, f is a function from $\{0,1\}^*$ to $\{0,1\}^*$, and x is an arbitrary input for which we would like to determine $f(x)$. We use r to denote a sequence of fair coin tosses; if $|x| = n$, then $|r| = w(n)$, where w is a polynomially bounded function of n.

Definition 2.1 *The function f is* **nonadaptively $k(n)$-random-self-reducible** *(abbreviated "nonadaptively k-rsr") if there are polynomial-time computable functions $\sigma_1, \ldots, \sigma_k$, and ϕ with the following properties.*

(1) For all n and all $x \in \{0,1\}^n$,

$$f(x) = \phi(x, r, f(\sigma_1(x,r)), \ldots, f(\sigma_k(x,r))),$$

for at least $3/4$ of all r's in $\{0,1\}^{w(n)}$, and

(2) For all n, all $\{x_1, x_2\} \subset \{0,1\}^n$, and all i, $1 \leq i \leq k$, if r is chosen uniformly at random, then $\sigma_i(x_1, r)$ and $\sigma_i(x_2, r)$ are identically distributed.

Feigenbaum, S. Kannan, and Nisan [14] use the term "$k(n)$-random-self-reduction" to describe a special case of Definition 2.1, i.e., the case in which each of the random variables $\sigma_i(x,r)$ is distributed uniformly over $\{0,1\}^n$.

Next we generalize Definition 2.1 to allow a multiround, adaptive strategy for choosing random queries.

Definition 2.2 *The function f is* **adaptively $k(n)$-random-self-reducible** *(abbreviated "adaptively k-rsr") if there is a probabilistic, polynomial-time oracle machine ϕ that, on input x of length n, produces $k(n)$* **rounds** *of f-oracle queries. The query $y_i(x,r)$ produced in round i may depend on all queries and answers in rounds 1 through $i-1$.*

The reduction ϕ must have the following properties.

(1) For all x, it outputs the correct answer $f(x)$ for at least $3/4$ of all $r \in \{0,1\}^{w(n)}$.

(2) For all n and all i, $1 \leq i \leq k$, if $|x_1| = |x_2| = n$ and r is chosen uniformly from $\{0,1\}^{w(n)}$, then the random variables $y_i(x_1, r)$ and $y_i(x_2, r)$ are identically distributed.
Note that condition (2) may not hold for $y_i(x,r)$ if wrong answers are given in earlier rounds.

We say that a function f is *poly-rsr* (or simply *rsr*) if there is some polynomially bounded function k such that f is either nonadaptively or adaptively k-rsr. The reductions themselves are also referred to as poly-rsr's or rsr's. A set S is poly-rsr if its characteristic function χ_S is poly-rsr.

Rsr's are special cases of *locally random reductions* (lrr's), which were introduced in [6] and defined formally in [7]. More precisely, a nonadaptive k-rsr for the function f is a $(1,k)$-lrr from f to f, in the terminology of [7].

The gist of Definitions 2.1 and 2.2 is that, for any fixed value of i, the distribution of random queries to the i^{th} oracle depends only on the length n of the input x. In keeping with the terminology in [1, 6], we say that an rsr "leaks at most n to each oracle." In cryptographic applications, it is often natural to consider reductions that leak at most some other function L; Definitions 2.1 and 2.2 have natural generalizations that fit these applications – see [1, 6] for details.

Lemma 2.3 *If a function f is nonadaptively (resp. adaptively) $k(n)$-rsr, then f is nonadaptively (resp. adaptively) $8t(n)k(n)$-rsr where condition (1) holds for at least $1 - 2^{-t(n)}$ of the r's in $\{0,1\}^{8t(n)w(n)}$.*

Proof: Let $r = r_1 \ldots r_{8t(n)}$ with each $r_i \in \{0,1\}^{w(n)}$. Define $\sigma_{ij} = \sigma_i(x, r_j)$ for $1 \leq i \leq k$ and $1 \leq j \leq 8t(n)$. Let $\phi_j = \phi(x, r_j, f(\sigma_{1j}), \ldots, f(\sigma_{kj}))$. Let the new ϕ choose the plurality of the ϕ_j's, handling ties arbitrarily. A simple application of Chernoff bounds [23, Lecture 4] gives us the lemma. ∎

We now recall some definitions and known results that will be used in Sections 3 and 4.

Let $f : \{0,1\}^* \to \{0,1\}$ be an arbitrary boolean function, and let $f_n : \{0,1\}^n \to \{0,1\}$ be the restriction of f to inputs $(x_1, \ldots, x_n) \in \{0,1\}^n$. For any finite field K_n, there is a unique multilinear polynomial $g_n \in K_n[X_1, \ldots, X_n]$ that *represents* f_n over K_n – i.e., g_n agrees with f_n on all inputs $(x_1, \ldots, x_n)$ in $\{0,1\}^n$. In the context of random-self-reducibility, we always take K_n to be a field of size at least $n+1$. The polynomial g_n has a standard explicit formula; we give the formula here and discuss some computational aspects of it in Section 4 below. Let $x = (x_1, \ldots, x_n)$ be an arbitrary element of K_n^n and $A = (a_1, \ldots, a_n)$ be an arbitrary element of $\{0,1\}^n$.

$$g_n(x) = \sum_{A \in \{0,1\}^n} \delta_A(x) f_n(A) \qquad (1)$$

$$\delta_A(x) = \prod_{i=1}^{n} (x_i - (1 - a_i))(-1)^{(1-a_i)} \qquad (2)$$

For $x \in \{0,1\}^n$, the monomial $\delta_A(x)$ is 1 if $A = x$, and it is 0 otherwise. We call g_n the *arithmetization of f_n over K_n* and $g = \{g_n\}_{n \geq 1}$ the arithmetization of f over $\{K_n\}_{n \geq 1}$. If $(x_1, \ldots, x_n)$ ranges over $\mathcal{Z}^n$, then g_n, as defined by Equations (1) and (2), is a polynomial in $\mathcal{Z}[X_1, \ldots, X_n]$, and g is called the *arithmetization of f over the integers.*

Fact 2.4 (the "low-degree polynomial trick") *If $g_n \in K_n[X_1, \ldots, X_n]$ has degree d_n and $|K_n| > d_n$, then $g = \{g_n\}_{n \geq 1}$ is nonadaptively $(d_n + 1)$-rsr. In particular, if g_n is the arithmetization over K_n of a boolean function f_n, then g is nonadaptively $(n+1)$-rsr.*

Proof: Let $\alpha_1, \ldots, \alpha_{d_n+1}$ be distinct elements of K_n. Choose coefficients $c_1, \ldots, c_n$ independently and uniformly at random from K_n, and let $\sigma_i(x_1, \ldots, x_n) = (c_1\alpha_i + x_1, \ldots, c_n\alpha_i + x_n)$ for $1 \leq i \leq d_n + 1$. Let

$$G(Z) = g_n(c_1 Z + x_1, \ldots, c_n Z + x_n).$$

Then G is a one-variable polynomial of degree at most d_n that satisfies

$$G(0) = g_n(x_1, \ldots, x_n).$$

The function ϕ of the rsr interpolates the $d_n + 1$ values $(\alpha_1, G(\alpha_1)), \ldots, (\alpha_{d_n+1}, G(\alpha_{d_n+1}))$ to recover the polynomial G and outputs the constant term. ∎

Beaver and Feigenbaum [6] created the low-degree polynomial trick in order to show that every function is locally-random-reducible to its arithmetization. Lipton [19] later observed that the same construction gives a random-*self*-reduction if the original function is itself a multivariate polynomial.

In [7], it is shown how to represent f_n as a degree-$(n/\log n)$ polynomial h_n over K_n by performing a simple change of variables. By Fact 2.4, the function $h = \{h_n\}_{n \geq 1}$ is nonadaptively $(1 + n/\log n)$-rsr.

Definition 2.5 *A complexity class C is* **#P-robust** *if* $\mathrm{FP}^C = \#\mathrm{P}^C$, *where* FP *denotes the class of all polynomial-time computable functions.*

In Section 4, we will use the following generalized version of #P.

Definition 2.6 (cf. [15]): *A function $f : \{0,1\}^* \to \mathcal{Z}$ is in the complexity class* **Gap-P** *if there is an* NP *machine M such that, for all x, $f(x)$ is the difference between the number of accepting computations of M on input x and the number of rejecting computations of M on input x. Equivalently, a function f is in* Gap-P *if it is the difference of two* #P *functions.*

By analogy with Definition 2.5, we have the following.

Definition 2.7 *A complexity class C is* **Gap-P-robust** *if* $\mathrm{FP}^C = \text{Gap-P}^C$.

Fact 2.8 *A complexity class C is* Gap-P*-robust if and only if it is #P-robust.*

Let $\langle \cdot, \cdot \rangle$ be a one-to-one, onto, polynomial-time computable, polynomial-time invertible pairing function from $\{0,1\}^* \times \{0,1\}^*$ to $\{0,1\}^*$. Gap-P has the following closure properties.

Fact 2.9 (cf. [15]): *If a function $f(\langle x,y\rangle) \in$* Gap-P *then the following functions are also in* Gap-P *for any polynomial p:*

1. $g(\langle x,y\rangle) = -f(\langle x,y\rangle)$
2. $g(x) = \sum_{|y|\leq p(|x|)} f(\langle x,y\rangle)$
3. $g(x) = \prod_{1\leq y\leq p(|x|)} f(\langle x,y\rangle)$

In particular, Gap-P is closed under subtraction.

Definition 2.10 *Let m be a positive integer greater than 1. A set S is in* $\mathrm{MOD}_m\mathrm{P}$ *if there is an* NP *machine M with the following property: If $x \in S$, then the number of accepting computations of M on input x is not equal to* 0 mod m*; if $x \notin S$, then the number of accepting computations of M on input x is equal to* 0 mod m.

Thus the class $\oplus$P, defined in [21], is $\mathrm{MOD}_2\mathrm{P}$ in the notation used here.

Fact 2.11 (cf. [8]): *If m_1 and m_2 are relatively prime, then $S \in \mathrm{MOD}_{m_1m_2}\mathrm{P}$ if and only if there are sets $S_1 \in \mathrm{MOD}_{m_1}\mathrm{P}$ and $S_2 \in \mathrm{MOD}_{m_2}\mathrm{P}$ such that $S = S_1 \cup S_2$.*

Fact 2.12 (cf. [8]): *If $p_1^{e_1}\cdots p_t^{e_t}$ is the prime factorization of m, then* $\mathrm{MOD}_m\mathrm{P} = \mathrm{MOD}_{p_1\cdots p_t}\mathrm{P}$.

We use the following class of straight-line programs of multivariate polynomials over $\mathcal{Z}$ to prove that sets complete for $\mathrm{MOD}_m\mathrm{P}$ are rsr.

Definition 2.13 (cf. [3]): A positive retarded arithmetic program with binary substitutions (PRAB) *is a sequence $P = \{p_1,\ldots,p_s\}$ of instructions such that, for every k, one of the following holds.*

(1) p_k is one of the constant polynomials 0 or 1.

(2) $p_k = x_i$ for some $i \leq k$.

(3) $p_k = 1 - x_i$ for some $i \leq k$.

(4) $p_k = p_i + p_j$ for some $i,j < k$.

(5) $p_k = p_ip_j$ for some $i + j \leq k$.

(6) $p_k = p_j(x_i = 0)$ or $p_j(x_i = 1)$ for some $i,j < k$. Here $p_j(x_i = \epsilon)$ refers to the polynomial obtained from p_j by replacing the variable x_i by the value ϵ.

We say that the program P **computes the polynomial** *p_s.*

Definition 2.14 (cf. [3]): *A sequence P_1, P_2, ... of PRAB's is* **uniform** *if there is a deterministic polynomial-time machine that, on input 1^n, outputs the instruction sequence P_n.*

Fact 2.15 (cf. [3]): *A set S is in* $\mathrm{MOD}_m\mathrm{P}$ *if and only if there is a uniform sequence $\{P_n\}_{n\geq 1}$ of PRAB's such that, for every $x \in \{0,1\}^*$,*

$$\chi_S(x) \equiv P_{|x|}(x) \bmod m.$$

We use $\mathrm{IP}(k)$ to denote the class of sets accepted by k-move interactive proof systems (cf. [5, 17]). Let $\mathrm{IP}(k)/poly$ be the class of sets accepted by k-move interactive proof systems in which the verifier is given polynomial-length advice in addition to probabilistic polynomial time.

Fact 2.16 (cf. [5, 18]): *For any constant k,* $\mathrm{IP}(k)/poly \subseteq \mathrm{NP}/poly$.

Finally we use the following known relationship between the levels of the PH and the corresponding nonuniform classes.

Fact 2.17 (cf. [27]): *If $\Sigma_i^P \subseteq \Pi_i^P/poly$, then the* PH *collapses to Σ_{i+2}^P.*

3 Complete Sets in the PH

Theorem 3.1 *If* SAT *is nonadaptively poly-rsr, then* $\overline{\mathrm{SAT}}$ *is in* $\mathrm{NP}/poly$.

Proof: It suffices to prove that the hypothesis implies that $\overline{\mathrm{SAT}}$ is in $\mathrm{IP}(2)/poly$. The conclusion then follows from Fact 2.16.

Let ϕ, σ_1, ..., σ_k, where $k = k(n)$ is a polynomially-bounded function, be a nonadaptive rsr for SAT. By Lemma 2.3, we can assume ϕ gives an incorrect value for the characteristic function of SAT with probability at most 2^{-n}.

Consider instances x of length n. The verifier's advice is the k-tuple $(p_1,\ldots,p_k)$, where p_i is the probability that a target instance $\sigma_i(x,r)$

is satisfiable. The probability is computed over all coin-toss sequences r.

The following is an IP(2)/*poly* protocol for $\overline{\text{SAT}}$. Let $m = 9k^3$. The quantifiers "for all $1 \leq i \leq k$" and "for all $1 \leq j \leq m$" are implicit whenever the subscripts i and j are used. For each $j \neq j'$, r_j is independent of $r_{j'}$.

Interactive proof system for $\overline{\text{SAT}}$:

V: Choose r_j and compute $y_{i,j} = \sigma_i(x, r_j)$.
$V \rightarrow P$: $\{y_{i,j}\}$.
$P \rightarrow V$: A claimed value $b_{i,j}$ for each $\chi_{SAT}(y_{i,j})$ and a witness for each claim that $b_{i,j} = 1$.
V: Accept iff
(1) $\phi(x, r_j, b_{1,j}, \ldots, b_{k,j}) = 0$, and
(2) More than $p_i m - 2\sqrt{km}$ of the $y_{i,j}$'s are satisfiable according to P.

Suppose that x is unsatisfiable and P is honest. Then acceptance condition (1) is met with probability at least $1 - m/2^n > 11/12$ for all $n > \log 12m$. We need only show condition (2) is met with probability at least 3/4 to have both conditions met with probability at least 2/3. Let $Z_{i,j}$ be an indicator variable that is 1 if $y_{i,j}$ is satisfiable and 0 otherwise, and let $Z_i = \sum_{j=1}^{m} Z_{i,j}$. Acceptance condition (2) is met if $Z_i > p_i m - 2\sqrt{km}$ for all i. Because $r_1, \ldots, r_m$ are pairwise independent, so are $Z_{i,1}, \ldots, Z_{i,m}$. Clearly $E(Z_i) = p_i m$ and $Var(Z_i) = p_i(1-p_i)m < m$. So Chebyshev's inequality suffices to show that

$$\begin{aligned} Prob(Z_i \leq p_i m - 2\sqrt{km}) &\leq Prob(|Z_i - p_i m| \geq 2\sqrt{km}) \\ &= Prob(|Z_i - E(Z_i)| \geq 2\sqrt{km}) \\ &\leq \frac{Var(Z_i)}{4km} < \frac{1}{4k}, \end{aligned}$$

for each i. Thus the probability that at least one Z_i is too small (i.e., the probability that condition (2) is not met) is at most 1/4.

Now suppose that x is satisfiable. We wish to show that the probability that V accepts is at most 1/3. If V accepts, condition (1) is satisfied, and so either

(a) Given correct answers $b_{i,j}$, ϕ says x is not satisfiable for some j, or

(b) P^* must have lied about the satisfiability of at least one $y_{i,j}$ for each j.

Event (a) can only happen with probability at most $m/2^n < 1/12$ for all $n > \log 12m$. Thus we need only show event (b) can happen with probability at most 1/4.

If P^* tells a total of m lies, there must be an i for which he claims that at least m/k satisfiable $y_{i,j}$'s are unsatisfiable. It suffices to show that, for each i, he can do this with probability at most $1/4k$ and still satisfy acceptance condition (2). This is another simple application of the Chernoff bounds in [23, Lecture 4]: The probability that P^* can tell m/k lies and still claim that more than $p_i m - 2\sqrt{km}$ of the $y_{i,j}$'s are satisfiable is just the probability that more than $p_i m + m/k - 2\sqrt{km}$ of the $y_{i,j}$'s are satisfiable. This probability is at most $e^{-2(m/k^2 - 4\sqrt{m/k} + 4k)} = e^{-2k} < 1/4k$ for $m = 9k^3$ and all positive integers k. ∎

Corollary 3.2 *If any* NP *set S is nonadaptively poly-rsr, then $\overline{S}$ is in* NP/*poly*.

Proof: Any set S in NP can be used in place of SAT in Theorem 3.1. ∎

Corollary 3.3 *If any* NP*-complete set is nonadaptively poly-rsr, then the* PH *collapses at the third level.*

Proof: This follows directly from Corollary 3.2 and Fact 2.17. ∎

Corollary 3.4 *If S is complete for Σ_i^p or Π_i^p, $i \geq 1$, and S is nonadaptively poly-rsr, then the* PH *collapses at the $(i+2)^{nd}$ level.*

Proof: The proofs of Theorem 3.1, Fact 2.17 and thus Corollary 3.3 relativize. If we relativize them with respect to an oracle O such that O is Σ_{i-1}^p-complete, we get Corollary 3.4. ∎

In the final version of our paper, we prove the following stronger result.

Theorem 3.5 *If* SAT *is adaptively $O(\log n)$-rsr, then $\overline{\text{SAT}}$ is in* NP/*poly*.

In fact, the conclusion that $\overline{\text{SAT}}$ is in NP/*poly* follows from the assumption that SAT has an adaptive rsr with $O(\log n)$ rounds of queries and polynomially many queries in each round. The proof technique used in Theorem 3.5 is very similar to that used in Theorem 3.1 here; unfortunately, this technique does not work unless the number of rounds is $O(\log n)$.

4 Complete Sets Above the PH

We first recall the following known positive result.

Theorem 4.1 *If f is #P-complete, then f is poly-rsr.*

Proof: This is essentially a restatement of a result in [19]. Let $PERM$ be the $\#P$-complete function that computes permanents of integer-matrices. To compute $PERM(x)$, it suffices to compute it modulo several primes and use the Chinese remainder theorem. Hence, PERM can be reduced to the computation of several low-degree polynomials over finite fields; it is thus rsr by Fact 2.4.

Let f be #P-complete. On input x of length n, the rsr for f proceeds as follows. Reduce x to one or more instances of PERM. Pad these instances if necessary so that their size depends only on n. Perform the standard rsr of PERM. The random PERM-instances thus produced can be mapped back to f-instances, because f is #P-complete. These f-instances leak at most n, because the random PERM-instances leak at most n. ∎

We now proceed to our new positive results. The first one is a straightforward extension of Theorem 4.1.

Theorem 4.2 *If S is complete for* PP, *then S is poly-rsr.*

Proof: The language classes P^{PP} and $\text{P}^{\#P}$ are equal. Thus S and PERM are ptime-equivalent. The rest of the proof is identical to that of Theorem 4.1. ∎

Theorem 4.3 *If a complexity class* C *is* #P*-robust, then complete sets for* C *are poly-rsr.*

Proof: By Fact 2.8, it suffices to show that Gap-P-robustness of C implies that complete sets for C are poly-rsr. Suppose that C is Gap-P-robust, and let S be a complete set for C. For each $n \geq 1$, let $f_n : \{0,1\}^n \to \{0,1\}$ be the characteristic function of S on strings of length n, and let $g = \{g_n\}_{n\geq 1}$ be the arithmetization of $f = \{f_n\}_{n\geq 1}$ over the integers. By Fact 2.9, we can compute g (using Equations (1) and (2)), in Gap-P^C and thus in FP^C. By Fact 2.4, g is (nonadaptively) $(n+1)$-rsr. Thus S, which is ptime-equivalent to g, is poly-rsr. ∎

Corollary 4.4 *Complete sets for* PSPACE *and* EXPTIME *are poly-rsr.*

Proof: This follows from the fact that PSPACE and EXPTIME are #P-robust. For example, let FPSPACE denote the set of functions computable in polynomial space. Then $\text{FP}^{\text{PSPACE}} \subseteq \#\text{P}^{\text{PSPACE}} \subseteq \text{FPSPACE}^{\text{PSPACE}}$, but $\text{FP}^{\text{PSPACE}} = \text{FPSPACE}^{\text{PSPACE}} = \text{FPSPACE}$, and thus $\text{FP}^{\text{PSPACE}} = \#\text{P}^{\text{PSPACE}}$. ∎

Note that is unknown whether #P is itself #P-robust.

Theorem 4.5 *If S is complete for* MOD_mP, *then S is poly-rsr. In particular, complete sets for $\oplus P$ are poly-rsr.*

Proof: By Facts 2.11 and 2.12, we can assume without loss of generality that m is prime.

Let S be a complete set in MOD_mP and $x = (x_1, \ldots, x_n)$ be an element of $\{0,1\}^n$ for which we would like to determine membership in S. By Fact 2.15, there is a PRAB $P_n = \{p_1, \ldots, p_s\}$ for which $\chi_s(x) = P_n(x) \bmod m$. Furthermore P_n can be generated in polynomial time, and it depends only on S and n (i.e., it leaks nothing about x except its length). We would like to apply Fact 2.4 (the low-degree polynomial trick) to p_s and then map the random p_s-instances back to S-oracle queries. However, there are only m distinct points in $\mathcal{Z}_m$, and the degree of p_s is a (polynomially bounded) function of n; thus,

there will not be enough interpolation points to recover $p_s(x)$ this way.

We deal with this problem as it is dealt with in the proof that $\mathrm{MOD}_m\mathrm{P}$-complete sets are checkable (cf. [3]). For every positive integer k, there is a unique finite field $\mathrm{GF}(m^k)$, and it is a vector space over $\mathcal{Z}_m$. Fix a basis for this vector space. We can represent each element a of $\mathrm{GF}(m^k)$ as the $k \times k$ matrix M_a denoting the linear transformation $x \mapsto ax$ of $\mathrm{GF}(m^k)$ to itself. Then M_0 is the zero matrix, M_1 is the identity matrix, $M_{a+b} = M_a + M_b$, and $M_{ab} = M_{ba} = M_a M_b$.

Choose k so that $m^k > d = degree(p_s)$, and represent the elements computed by P_n as matrices over $\mathcal{Z}_m$. There is another PRAB, say $\{p_1(1,1), \ldots, p_1(k,k), \ldots, p_s(1,1), \ldots, p_s(k,k)\}$, where $p_i(r,c)$ computes the element in row r, column c of $p_i(x_1, \ldots, x_n)$. Because $m^k > d$, we can apply Fact 2.4 to p_s as follows. Let $\alpha_1, \ldots, \alpha_{d+1}$ be distinct elements of $\mathrm{GF}(m^k)$. Choose $c_1, \ldots, c_n$ independently and uniformly at random from the set of all $k \times k$ matrices over $\mathcal{Z}_m$ that encode elements of $\mathrm{GF}(m^k)$. Then $P_n(c_1 Z + x_1, \ldots, c_n Z + x_n)$ is a degree-d, one-variable polynomial with constant term $P_n(x) = \chi_s(x)$. So evaluate P_n at the $d+1$ uniformly distributed inputs $(c_1\alpha_1 + x_1, \ldots, c_n\alpha_1 + x_n), \ldots, (c_1\alpha_{d+1} + x_1, \ldots, c_n\alpha_{d+1} + x_n)$ and interpolate.

It remains to show that the computation of $P_n(c_1\alpha_j + x_1, \ldots, c_n\alpha_j + x_n)$ can be reduced in polynomial time to a sequence of S-oracle queries. In the matrix representation of P_n, each $p_i(r,c)$ is an instruction in a PRAB over $\mathcal{Z}_m$. Thus it is polynomial-time reducible to S by Fact 2.15, because S is complete for $\mathrm{MOD}_m\mathrm{P}$.

As in the previous proofs in this section, each S-oracle call leaks at most n, because each random input $(c_1\alpha_j + x_1, \ldots, c_n\alpha_j + x_n)$ leaks at most n. ∎

5 Open Problems

Open problems abound, including:

- Do NP-complete sets have adaptive k-rsr's for some $k \gg \log n$?
- Are NP-complete sets checkable in the sense of [9]? Note that all known checkers for sets that are complete for natural complexity classes use rsr's.
- What other sets do or do not have rsr's? How about incomplete sets? Sets and functions complete for classes C that satisfy $\mathrm{PH} \subseteq \mathrm{BPP}^C$ and $\mathrm{P}^C \subseteq \mathrm{PSPACE}$? The classes $\mathrm{MOD}_m\mathrm{P}$, PP, and $\#P$ all fall between the PH and PSPACE in this sense (cf. [24]).

6 Acknowledgments

We thank Manuel Blum, Russell Impagliazzo, Steven Rudich, and Gábor Tardos for their comments on an earlier version of this paper.

References

[1] M. Abadi, J. Feigenbaum, and J. Kilian. On Hiding Information from an Oracle, *J. Comput. System Sci.* 39 (1989), 21–50.

[2] L. Babai. Random Oracles Separate PSPACE from the Polynomial-time Hierarchy, *Inf. Proc. Letters* 26 (1987), 51–53.

[3] L. Babai and L. Fortnow. A Characterization of $\#P$ by Straight Line Programs of Polynomials, with Applications to Interactive Proofs and Toda's Theorem, *Proc. of the 31st FOCS* (1990), IEEE, 26–35.

[4] L. Babai, L. Fortnow, and C. Lund. Nondeterministic Exponential Time has Two-Prover Interactive Protocols, *Proc. of the 31st FOCS* (1990), IEEE, 16–25.

[5] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes, *J. Comput. System Sci.* 36 (1988), 254–276.

[6] D. Beaver and J. Feigenbaum. Hiding Instances in Multioracle Queries, *Proc. of the 7th STACS* (1990), Springer Verlag LNCS 415, 37–48.

[7] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with Low Communication Overhead, *Proc. of the 10th CRYPTO* (1990), Springer Verlag LNCS, to appear.

[8] R. Beigel, J. Gill, and U. Hertrampf. Counting Classes: Thresholds, Parity, Mods, and Fewness, *Proc. of the 7th STACS* (1990), Springer Verlag LNCS 415, 49–57.

[9] M. Blum and S. Kannan. Designing Programs that Check Their Work, *Proc. of the 21st STOC* (1989), ACM, 86–97.

[10] M. Blum, M. Luby, and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems, *Proc. of the 22nd STOC* (1990), ACM, 73–83.

[11] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudorandom Bits, *SIAM J. Comput.* 13 (1984), 850–864.

[12] J. Cai. With Probability One, a Random Oracle Separates PSPACE From the Polynomial-Time Hierarchy, *J. Comput. System Sci.* 38 (1989), 68–85.

[13] J. Feigenbaum and L. Fortnow. On the Random-Self-Reducibility of Complete Sets, University of Chicago Technical Report 90-22, Computer Science Department, August 20, 1990.

[14] J. Feigenbaum, S. Kannan, and N. Nisan. Lower Bounds on Random-Self-Reducibility, *Proc. of the 5th Structures* (1990), IEEE, 100-109.

[15] S. Fenner, L. Fortnow, and S. Kurtz. Gap-Definable Counting Classes, these proceedings.

[16] S. Goldwasser and S. Micali. Probabilistic Encryption, *J. Comput. System Sci.* 28 (1984), 270–299.

[17] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems, *SIAM J. Comput.* 18 (1989), 186–208.

[18] S. Goldwasser and M. Sipser. Public Coins vs. Private Coins in Interactive Proof Systems, *Advances in Computing Research - Vol. 5: Randomness and Computation*, JAI Press, Greenwich, 1989, 73–90.

[19] R. Lipton. New Directions in Testing, in *Distributed Computing and Cryptography*, AMS/-ACM Series on Discrete Mathematics and Theoretical Computer Science, 2 (1991), 191–202.

[20] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems, *Proc. of the 31st FOCS* (1990), IEEE, 2–10.

[21] C. Papadimitriou and S. Zachos. Two remarks on the complexity of counting, *Proc. of the 6th GI Conf. on Theoretical Computer Science* (1983), Springer Verlag LNCS 145, 269–276.

[22] A. Shamir. IP = PSPACE, *Proc. of the 31st FOCS* (1990), IEEE, 11–15.

[23] J. Spencer. *Ten Lectures on the Probabilistic Method*, CBMS 52, SIAM, Philadelphia, 1987, page 29.

[24] S. Toda. On the Computational Power of PP and $\oplus$P, *Proc. of the 30th FOCS* (1989), IEEE, 514–519.

[25] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information, *Proc. of the 28th FOCS* (1987), IEEE, 472–482.

[26] L. Valiant. The Complexity of Computing the Permanent, *Theor. Comput. Sci.* 8 (1979), 189–201.

[27] C. Yap. Some Consequences of Nonuniform Conditions on Uniform Classes, *Theor. Comput. Sci.* 26 (1983), 287–300.