ON THE RANDOM-SELF-REDUCIBILITY OF COMPLETE SETS*

JOAN FEIGENBAUM† AND LANCE FORTNOW‡

Abstract. 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 nonadaptively random-self-reducible, unless the hierarchy collapses. In particular, NP-complete sets are not nonadaptively 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.

Key Words. random-self-reductions, complexity classes, interactive proof systems, program checkers

AMS(MOS) subject classification. 68Q05, 68Q15

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, $f(y_1), \ldots, f(y_k)$, 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 randomized 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, it follows from a result in [21] that the PERM (permanent of integer matrices) function is random-self-reducible. The PERM function is also #P-complete (cf. [29]); thus, if PERM could be computed efficiently *on average* (with respect to the target distribution of the reduction), then *every* function in #P could, with a randomized algorithm, 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 [3] to obtain a simple proof that a random oracle separates the polynomial hierarchy (PH) from PSPACE. (An earlier proof of this result in [14] does not use random-self-reducibility.)

Interactive proof systems and program checkers: Random-self-reductions are crucial ingredients in many of the original examples of interactive proof systems and program checkers (cf. [11, 18]). 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

* This work was supported in part by National Science Foundation grant CCR 90-09936. This is a revised version of the paper "On the Random-Self-Reducibility of Complete Sets," appearing in Proceedings of the 6th Annual Structure in Complexity Theory Conference, June 30–July 3, 1991, Chicago, Illinois, ©1991 by IEEE Computer Society.

† AT&T Bell Laboratories, Room 2C473, 600 Mountain Avenue, P. O. Box 636, Murray Hill, NJ 07974-0636, jf@research.att.com

‡ University of Chicago, Computer Science Department, 1100 East 58th Street, Chicago, IL 60637, fortnow@cs.uchicago.edu

formally in [12, 22, 28]. These ideas play a crucial role in the characterization of the language-recognition power of interactive proof systems (cf. [5, 22, 25]). 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. [17]) and *cryptographically strong pseudorandom number generation* (cf. [13]). Random-self-reductions also provide natural examples of *instance-hiding schemes* (cf. [1, 7, 8]), in which a weak, private computing device uses the resources of a powerful, shared computing device without revealing its private data.

Although random-self-reducibility had been used for a long time in the design and analysis of cryptographic protocols (cf., e.g., [17, 13]), it was first defined formally and studied from a complexity theoretic point of view 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 polynomial-time hierarchy collapses at the third level.

Random-self-reductions that produce several, correlated random instances $y_1, \ldots, y_k$ were defined formally by Feigenbaum, Kannan, and Nisan [15]; 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 polynomial-time hierarchy 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, 7, 8]. 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 [7, 8] 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 [15]. 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 polynomial-time hierarchy 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 polynomial-time hierarchy collapses at the third level. This strengthens the main result in [15].
- 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 polynomial-time hierarchy. 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.

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. The number of random queries produced by a reduction, denoted $k(n)$, is also a polynomially bounded function of n.

DEFINITION 2.1. *A function f is* **nonadaptively $k(n)$-random-self-reducible** *(abbreviated "nonadaptively k-rsr") if there are polynomial-time computable functions σ 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, Kannan, and Nisan [15] 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) is not required to 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.

Locally random reductions (lrr's) are a generalization of nonadaptive random-self-reductions. In a (t,k)-lrr from f to g, an instance x in the domain of f is mapped (nonadaptively) to k instances $y_1, \ldots, y_k$ in the domain of a different function g. For any $\{i_1, \ldots, i_t\} \subseteq \{1, \ldots, k\}$, the distribution induced on target queries $y_{i_1}, \ldots, y_{i_t}$ is the same for input instances x_1 and x_2 if $|x_1| = |x_2|$. Thus a nonadaptive k-rsr for f is a $(1,k)$-lrr from f to f. *Instance-hiding schemes* (ihs's) provide a further generalization of this notion. In a t-private, k-oracle ihs for f, the querier may use a multiround, adaptive strategy to query k physically separated, arbitrarily powerful oracles; the oracles may also use an adaptive strategy and may flip coins. The *view* of any set of at most t of the oracles (i.e., the transcript of queries and answers together with the coin flips of the oracles) depends only on the length of the input instance. One-oracle ihs's were studied by Abadi, Feigenbaum, and Kilian [1], who showed that NP-hard functions do not have one-oracle ihs's unless the polynomial-time hierarchy collapses at the third level. The question of whether multioracle ihs's exist was posed by Rivest [1] and answered by Beaver and Feigenbaum [7]: Every function f has a 1-private, $(n+1)$-oracle ihs. In fact, the general ihs construction of [7] uses only one round of queries and does not require the oracles to flip coins; so, in current terminology, it is a $(1, n+1)$-lrr. The *term* lrr was subsequently introduced and formally defined by Beaver, Feigenbaum, Kilian, and Rogaway [8], who also gave an improvement of the Beaver-Feigenbaum construction: For every polynomially bounded $t = t(n)$ and every function f, there is a function g such that f is $(t, (tn/\log n) + 1)$-lrr to g.

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, 7], 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, 7] for details.

In several proofs, we will use the following Chernoff bounds on the binomial distribution, which are taken directly from [26, Lecture 4, p. 29].

FACT 2.3. *Let $Y_1, \ldots, Y_n$ be independent with $Pr(Y_j = 1) = p_j$ and $Pr(Y_j = 0) = 1 - p_j$, and normalize by setting $X_j = Y_j - p_j$. Set $p = (p_1 + \cdots p_n)/n$ and $X = X_1 + \cdots X_n$. Then*

$$Pr(X > a) < e^{-2a^2/n} \tag{1}$$

and

$$Pr(X < -a) < e^{-a^2/2pn} \tag{2}$$

LEMMA 2.4. *If a function f is nonadaptively (resp. adaptively) $k(n)$-rsr, then f is nonadaptively (resp. adaptively) $24t(n)k(n)$-rsr where condition (1) holds for at least $1 - 2^{-t(n)}$ of the r's in $\{0,1\}^{24t(n)w(n)}$.*

Proof. Let $r = r_1 \ldots r_{24t(n)}$ with each $r_i \in \{0,1\}^{w(n)}$. Define $\sigma_{ij} = \sigma(i, x, r_j)$ for $1 \le i \le k$ and $1 \le j \le 24t(n)$. 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. Now apply Inequality (2) from Fact 2.3 with $p = 3/4$, $n = 24t$, $a = -n/4$, and $Y_j = 1$ if and only if $\phi_j = f(x)$. □

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 $y = (y_1, \ldots, y_n)$ be an arbitrary element of $\{0,1\}^n$.

$$g_n(x) = \sum_{y \in \{0,1\}^n} \delta_y(x) f_n(y) \tag{3}$$

$$\delta_y(x) = \prod_{i=1}^{n} (x_i - (1 - y_i))(-1)^{(1-y_i)} \tag{4}$$

For $x \in \{0,1\}^n$, the monomial $\delta_y(x)$ is 1 if $y = x$, and it is 0 otherwise. We call g_n the *arithmetization of f_n over K_n* and $g = \{g_n\}_{n \ge 1}$ the arithmetization of f over $\{K_n\}_{n \ge 1}$.

FACT 2.5. *(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 $g = \{g_n\}_{n \ge 1}$ 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 $1 \le i \le d_n + 1$. 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 $(\alpha_1, G(\alpha_1)), (\alpha_2, G(\alpha_2)), \ldots, (\alpha_{d_n+1}, G(\alpha_{d_n+1}))$ to recover the polynomial G and outputs the constant term. □

The random-self-reducibility of multivariate polynomials is the key to some of the results stated above. Beaver and Feigenbaum's general construction of multioracle ihs's [7], which they described in terms of arithmetic circuits, can be described in current terminology as follows: Every boolean function is $(1, n+1)$-lrr to its arithmetization over $\{K_n\}_{n\geq 1}$, where K_n is any finite field of size greater than n. Lipton [21] later used the same construction to show that multivariate polynomials are, in his terms, *randomly testable*; his was the first paper to state the construction in terms of polynomials instead of arithmetic circuits. In current terminology, Lipton's observation is that multivariate polynomials are nonadaptively rsr, provided the degree is polynomially bounded in the number of variables. Lipton also pointed out that the function that computes the permanent of a matrix over a finite field is a low-degree multivariate polynomial and thus randomly testable. Finally, Beaver, Feigenbaum, Kilian, and Rogaway [8] showed how to represent f_n as a degree-$(n/\log n)$ polynomial h_n over K_n by performing a simple change of variables, thus obtaining a $(1, (n/\log n)+1)$-lrr from f to h.

DEFINITION 2.6. *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 2.7. *(cf. [16]): 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 2.6, we have the following.

DEFINITION 2.8. *A complexity class C is* **Gap-P-robust** *if* $\mathrm{FP}^C = \text{Gap-P}^C$.

FACT 2.9. *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 2.10. *(cf. [16]): 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 2.11. *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* $0 \bmod m$*; if $x \notin S$, then the number of accepting computations of M on input x is equal to* $0 \bmod m$.

Thus the class $\oplus$P, defined in [23], is $\mathrm{MOD}_2\mathrm{P}$ in the notation used here.

FACT 2.12. *(cf. [9, 20]): If m_1 and m_2 are relatively prime, then $S \in \mathrm{MOD}_{m_1 m_2}\mathrm{P}$ 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 2.13. *(cf. [9, 20]): 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 2.14. *(cf. [4]): 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) $p_k = p_i p_j$ *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 2.15. *(cf. [4]): 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 2.16. *(cf. [4]): 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 AM^{poly} to denote the class of sets accepted by bounded-round Arthur-Merlin games (cf. [6]) in which Arthur is given polynomial-length advice in addition to probabilistic polynomial time. Note that this class is not necessarily the same as $\mathrm{AM}/poly$, because AM^{poly} requires proper probabilities of acceptance only when the advice is correct. Because the main results of [6, 19] relativize, we have:

FACT 2.17. $\mathrm{AM}^{poly} = \mathrm{NP}/poly$*.*

Finally we use the following known relationship between levels of the polynomial-time hierarchy and the corresponding nonuniform classes.

FACT 2.18. *(cf. [30]): If* $\Sigma_i^P \subseteq \Pi_i^P/poly$*, then the polynomial-time hierarchy collapses to* Σ_{i+2}^P*. This fact relativizes: For any* O*, if* $\Sigma_i^{P,O} \subseteq \Pi_i^{P,O}/poly$*, then* $\mathrm{PH}^O \subseteq \Sigma_{i+2}^{P,O}$*.*

3. Complete Sets in the Polynomial-Time Hierarchy.

THEOREM 3.1. *If* S *is in* NP *and is nonadaptively poly-rsr, then* $\overline{S}$ *is in* AM^{poly}*.*

Proof. Let σ, ϕ be a nonadaptive k-rsr for S, where $k = k(n)$ is a polynomially bounded function. By Lemma 2.4, we can assume ϕ gives an incorrect value for the characteristic function of S 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 in S. The probability is computed over all coin-toss sequences r.

We denote by $Trans(x, r)$ the *transcript* of the reduction σ, ϕ on input x and random string r. That is, if $y_i = \sigma(i, x, r)$ and $b_i = \chi_S(y_i)$, then $Trans(x, r) = (y_1, b_1, \ldots, y_k, b_k)$. Fix a specific NP machine M that accepts S. Let $ATrans(x, r)$, an *augmented transcript*, be $(y_1, b_1, w_1, \ldots, y_k, b_k, w_k)$, where y_i and b_i are as before, $w_i =$ NIL if $b_i = 0$, and w_i is a witness, with respect to M, that $y_i \in S$ if $b_i = 1$.

The following is an AM^{poly} protocol for $\overline{S}$. Let $m = 9k^3$.

Interactive proof system for $\overline{\mathbf{S}}$:

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 $r_{j'}$.

V: Choose r_j.
$V \to P$: $\{r_j\}$.
$P \to V$: A claimed value $(y_{1,j}, b_{1,j}, w_{1,j}, \ldots, y_{k,j}, b_{k,j}, w_{k,j})$ for $ATrans(x, r_j)$.
V: Accept iff

(1) $\phi(x, r_j, b_{1,j}, \ldots, b_{k,j}) = 0$,

(2) More than $p_i m - 2\sqrt{km}$ of the $y_{i,j}$'s are in S according to P, and

(3) If $w_{i,j} \neq$ NIL, then it is a correct witness that $y_{i,j} \in S$.

Suppose that x is not in S and P is honest. Then acceptance condition (1) is met with probability at least $1 - m/2^n > 11/12$ for all $n > \log 12m$. Condition (3) is of course always met if P is honest. We need only show that condition (2) is met with probability at least 3/4 to have all three conditions met with probability at least 2/3. Let $Z_{i,j}$ be an indicator variable that is 1 if $y_{i,j}$ is in S 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 in S. 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 in S for some j, or

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

(The optimal cheating prover would never violate condition (3).) Event (a) can only happen with probability at most $m/2^n < 1/12$ for all $n > \log 12m$. Thus we need only show that 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 of the $y_{i,j}$'s that are in S are not in S. It suffices to show that, for each i, he can do this with probability at most $1/4k$ and still satisfy acceptance condition (2). 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 in S is just the probability that more than $p_i m + m/k - 2\sqrt{km}$ of the $y_{i,j}$'s are in S. 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. We obtain this bound by using Inequality (1) from Fact 2.3, with $a = m/k - 2\sqrt{km}$ and $n = m$. □

The technique of showing that a particular type of random-self-reduction for S implies a type of interactive proof system for $\overline{S}$ was first used by Feigenbaum, Kannan, and Nisan [15, Theorem 4.4]. There it is shown that if S has what they call a "one-sided 1-rsr," then $\overline{S} \in \mathrm{AM}^{poly}$. These reductions are much more restricted than the type of rsr's considered here; a precise definition can be found in [15].

In fact, the following stronger statement can be made. This observation is due to Mario Szegedy.

COROLLARY 3.2. *If S is in* NP *and is nonadaptively poly-rsr, then $\overline{S}$ is in* AM^{log}.

Proof. Let σ, ϕ be a nonadaptive rsr for S. Define a new nonadaptive rsr σ', ϕ' as follows. On input x and random string r', first choose a uniformly random permutation π on $\{1, \ldots, k\}$. Let r be the unused portion of r'. For $1 \leq i \leq k$, let $\sigma'(i, x, r') \equiv \sigma(\pi(i), x, r)$. Let $\phi'(x, r', b_1, \ldots, b_k) \equiv \phi(x, r, b_{\pi^{-1}(1)}, \ldots, b_{\pi^{-1}(k)})$. Now all of the random variables $\sigma'(1, x, r'), \ldots, \sigma'(k, x, r')$ are identically distributed. The proof system for $\overline{S}$ is essentially the same as the one given in the proof of Theorem 3.1. The probabilities $p_1, \ldots, p_k$ are all equal, because $\sigma'(1, x, r'), \ldots, \sigma'(k, x, r')$ are identically distributed. Let $p = p_1 = \cdots = p_k$. The integer $\lceil pm - 2\sqrt{km} \rceil$ is sufficient advice for the verifier on inputs of length n, and it can be written down in $O(\log n)$ bits. □

COROLLARY 3.3. *If any* NP*-complete set is nonadaptively poly-rsr, then the polynomial-time hierarchy collapses at the third level.*

Proof. This follows directly from Theorem 3.1 and Facts 2.17 and 2.18. □

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 polynomial-time hierarchy collapses at the $(i+2)^{nd}$ level.*

Proof. The proofs of Theorem 3.1, Fact 2.18 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. □

We end this section with a partial negative result about adaptive rsr's for NP-complete sets.

THEOREM 3.5. *If S is* NP*-complete and is adaptively $O(\log n)$-rsr, then the polynomial-time hierarchy collapses at the third level.*

Proof (sketch): We give the structure of the proof in some detail but omit the probability calculations. All of the calculations involve Chernoff bounds and are analogous to the ones used in the proof of Theorem 3.1.

As in the nonadaptive case, we will show that the hypothesis implies that $\overline{S} \in \mathrm{AM}^{poly}$. Suppose that S is adaptively k-rsr, where $k(n) = O(\log n)$. In the augmented transcript $(y_1, b_1, w_1, \ldots, y_k, b_k, w_k)$ of an adaptive reduction, y_i denotes the oracle query asked in the i^{th} round.

The AM^{poly} proof system for $\overline{S}$ is almost identical to the one in Theorem 3.1. The only differences are that we may have to use a different polynomial for m and that the verifier, in acceptance condition (3), must also check that $y_{i,j}$ is indeed the query that the reduction would produce in round i if the input string is x, the random string is r_j, and the answers in rounds 1 through $i-1$ are $b_{1,j}$ through $b_{i-1,j}$. However, it is trickier to show that the proof system is correct in the adaptive case. The problem arises when x is in S and P^* must lie about at least one $b_{i,j}$ for each j. In the nonadaptive case, the only way that P^* can lie is to claim that $y_{i,j}$ is not in S when it really is. In the adaptive case, this is not necessarily true.

Consider a k by m matrix whose (i,j) entry is P^*'s claimed value for $b_{i,j}$. In order to convince V to accept an input x that is really in S, P^* must claim that the number of 1's in row i is greater than $p_i m - 2\sqrt{km}$, and he must *spoil* every column – i.e., he must lie about at least one $b_{i,j}$ for each j. The problem is that there is a tradeoff between these two requirements that could work to P^*'s advantage. Suppose that $b_{i_0,j}$ is the first lie that P^* tells in column j. (If this is so, we say that "column j is spoiled in row i_0.") For this first lie, it must be the case that $y_{i,j}$ is in S but P^* claims it isn't. However, this incorrect $b_{i_0,j}$ is used in the subsequent computation of queries $y_{i,j}$; this may produce $y_{i,j}$'s, $i > i_0$, that are in S where the correct value of $b_{i_0,j}$ would have produced $y_{i,j}$'s that are not in S.

For $k = O(\log n)$, we can still choose $m = poly(n)$ so that the proof system works. We make the following worst-case assumption in our argument: If column j is spoiled in row i_0, then the rest of the column, i.e., all $y_{i,j}$ with $i > i_0$, consists entirely of elements of S. We now estimate now many columns can be spoiled in each row.

The expected number of $y_{1,j}$'s that are in S is $p_1 m$; so, with very high probability, for these particular choices $r_1, \ldots, r_m$, the actual number is less than $p_1 m + 2\sqrt{km}$. P^* must claim that more than $p_1 m - 2\sqrt{km}$ are in S; so, with high probability, he may spoil at most $4\sqrt{km}$ columns in row 1 without getting caught.

What happens in row 2? With high probability, the actual number of $y_{2,j}$'s in S is less than $p_2 m + 2\sqrt{km}$. We assume that the columns spoiled in row 1 contribute $4\sqrt{km}$ additional $y_{2,j}$'s in S.

P^* must claim that more than $p_2m - 2\sqrt{km}$ are in S; so, with high probability, he may spoil at most $8\sqrt{km}$ new columns in row 2.

In row 3, the actual number of $y_{3,j}$'s in S is less than $p_3m + 2\sqrt{km}$, with high probability. Columns spoiled in rows 1 and 2 contribute at most $12\sqrt{km}$ additional $y_{3,j}$'s in S. Thus P^* may, with high probability, spoil at most $16\sqrt{km}$ new columns in row 3 and still claim that $p_3m - 2\sqrt{km}$ of the $y_{3,j}$'s are in S.

Continuing in this manner, we see that, with high probability, at most $2^{k+2}\sqrt{km}$ columns are spoiled in all k rows. For $k \leq c\log n$, we can choose $m = n^{4c}$, say, and prevent P^* from spoiling all of the columns.

In fact, the conclusion that $\overline{S}$ is in AM^{poly} follows from the weaker assumption that S has an adaptive rsr with $O(\log n)$ rounds of queries and polynomially many queries in each round. Unfortunately, this proof technique does not work unless the number of rounds is $O(\log n)$. ▮

4. Complete Sets Above the Polynomial-Time Hierarchy. We first recall the following known positive result.

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

Proof. This follows easily from the results on low-degree polynomials discussed in Section 2 above. Let PERM be the $\#P$-complete function that computes permanents of integer-matrices. An instance x of PERM can be reduced to the computation of $\mathrm{PERM}(x) \bmod p_i$, for some small collection of primes p_i – to recover $\mathrm{PERM}(x)$ from $\{\mathrm{PERM}(x) \bmod p_i\}$, use the Chinese Remainder Theorem. For each p_i, $\mathrm{PERM}(x) \bmod p_i$ is just a low-degree polynomial over a finite field. Thus, by Fact 2.5, it can be reduced to the evaluation of a small collection of random instances $\{\mathrm{PERM}(y_{ij}) \bmod p_i\}$. These y_{ij}'s can be regarded as random instances of PERM – from the value of $\mathrm{PERM}(y_{ij})$ over the integers, $\mathrm{PERM}(y_{ij}) \bmod p_i$ can be found simply by reducing mod p_i. In summary, the mapping from x to $\{y_{ij}\}$ is an rsr for PERM.

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: For any $l \geq k$, a k-by-k matrix M can be "padded" out to an l-by-l matrix M' with the same permanent by letting $M'(i,j) = M(i,j)$, for $1 \leq i,j \leq k$, $M'(i,i) = 1$, for $k < i \leq l$, and $M'(i,j) = 0$, for all other values of i and j. Perform the above 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.

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

Proof. It is well known that the language classes P^{PP} and $\mathrm{P}^{\#P}$ 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 2.9, 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 p_n be a prime greater than n, $f_n : \{0,1\}^n \to \{0,1\}$ be the characteristic function of S on strings of length n, and $g = \{g_n\}_{n\geq 1}$ be the arithmetization of $f = \{f_n\}_{n\geq 1}$ over $\{\mathrm{GF}(p_n)\}_{n\geq 1}$. By Fact 2.10, we can compute g (using Equations (3) and (4)), in Gap-P^C and thus in FP^C. By Fact 2.5, g is (nonadaptively) $(n+1)$-rsr. On input x, the random-self-reduction of S proceeds as follows: Generate p_n; interpret the input instance x as an element of $Dom(g)$; apply the low-degree polynomial trick to get random instances $y_1, \ldots, y_{n+1}$; reduce the computation of $g(y_i)$ to membership queries about S, which can be done because $g \in \mathrm{FP}^C$ and S is complete for C. The entire reduction leaks at most n, because each of its components leaks at most n. □

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

$$\mathrm{FP}^{\mathrm{PSPACE}} \subseteq \#\mathrm{P}^{\mathrm{PSPACE}} \subseteq \mathrm{FPSPACE}^{\mathrm{PSPACE}} = \mathrm{FPSPACE}$$

and thus $\mathrm{FP}^{\mathrm{PSPACE}} = \#\mathrm{P}^{\mathrm{PSPACE}}$. □

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

THEOREM 4.5. *If S is complete for* $\mathrm{MOD}_m\mathrm{P}$*, then S is poly-rsr. In particular, complete sets for $\oplus P$ are poly-rsr.*

Proof. By Facts 2.12 and 2.13, we can assume without loss of generality that m is prime. The reduction to the case of squarefree m is trivial, by Fact 2.13. If $m = m_1 \cdots m_t$, where the m_i's are distinct primes, then Fact 2.12 tells us that $S = S_1 \cup \cdots \cup S_t$, where $S_i \in \mathrm{MOD}_{m_i}\mathrm{P}$. Thus, a query about membership of x in S reduces to the disjunction of queries about membership of x in S_1, …, S_t. In what follows, we will show that S_i is rsr. Suppose that y is a random query produced by the rsr for S_i on input x. We must show how to compute $\chi_{S_i}(y)$ by making queries to an S oracle. First note that there is a set T_i in $\mathrm{MOD}_m\mathrm{P}$ such that $\chi_{S_i}(z) = \chi_{T_i}(z)$ for all z: If the underlying NP machine for S_i is M_i, then the underlying NP machine for T_i is M_i', where each computation path (accepting or rejecting) in M_i is replaced by m/m_i distinct paths in M_i'. Finally, T_i can be reduced to S, because S is complete for $\mathrm{MOD}_m\mathrm{P}$.

Let S be a complete set in $\mathrm{MOD}_m\mathrm{P}$, where m is prime, 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 2.16, 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 2.5 (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. [4]). 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. This entails finding a polynomial of degree k that is irreducible over $\mathcal{Z}_m$, which can be done in probabilistic polynomial time [10, 24]. (In fact, we could choose k so that all that is required is a polynomial of degree l, where $\Omega(k/\log m) = l \leq k$, that is irreducible over $\mathcal{Z}_m$. Such a polynomial could be generated in deterministic polynomial time [2], but this is not necessary for our purpose, which is to use the polynomial in a reduction that is inherently probabilistic.) 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 2.5 to p_s as follows. Let α_1, …, α_{d+1} be distinct elements of $\mathrm{GF}(m^k)$. Choose c_1, …, 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 $(c_1\alpha_1 + x_1, \ldots, c_n\alpha_1 + x_n)$, …, $(c_1\alpha_{d+1} + x_1, \ldots, c_n\alpha_{d+1} + x_n)$ 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 2.16, 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 [11]? 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 PH and PSPACE in this sense (cf. [27]).

Acknowledgments. We would like to thank Manuel Blum, Russell Impagliazzo, Steven Rudich, Gábor Tardos, and the referee for their comments on earlier versions of this paper.

REFERENCES

[1] M. Abadi, J. Feigenbaum, and J. Kilian, *On hiding information from an oracle*, Journal of Computer and System Sciences, 39 (1989), pp. 21–50.

[2] L. Adleman and H. Lenstra, *Finding irreducible polynomials over finite fields*, in Proceedings of the 16th Symposium on the Theory of Computing, ACM, New York, 1986, pp. 350–355.

[3] L. Babai, *Random oracles separate PSPACE from the polynomial-time hierarchy*, Information Processing Letters, 26 (1987), pp. 51–53.

[4] L. Babai and L. Fortnow, *Arithmetization: A new method in structural complexity theory*, Computational Complexity, 1 (1991), pp. 41–66.

[5] L. Babai, L. Fortnow, and C. Lund, *Non-deterministic exponential time has two-prover interactive protocols*, Computational Complexity, 1 (1991), pp. 3–40.

[6] L. Babai and S. Moran, *Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes*, Journal of Computer and System Sciences, 36 (1988), pp. 254–276.

[7] D. Beaver and J. Feigenbaum, *Hiding instances in multioracle queries*, in Proceedings of the 7th Symposium on Theoretical Aspects of Computer Science, vol. 415 of Lecture Notes in Computer Science, Springer, Berlin, 1990, pp. 37–48.

[8] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, *Security with low communication overhead*, in Advances in Cryptology – Crypto '90, vol. 537 of Lecture Notes in Computer Science, Springer, Berlin, 1991, pp. 62–76.

[9] R. Beigel and J. Gill, *Counting classes: Thresholds, parity, mods, and fewness*, Theoretical Computer Science, 103 (1992), pp. 3–23.

[10] E. Berlekamp, *Algebraic Coding Theory*, McGraw Hill, New York, 1968.

[11] M. Blum and S. Kannan, *Designing programs that check their work*, in Proceedings of the 21st Symposium on the Theory of Computing, ACM, New York, 1989, pp. 86–97.

[12] M. Blum, M. Luby, and R. Rubinfeld, *Self-testing/correcting, with applications to numerical problems*, in Proceedings of the 22nd Symposium on the Theory of Computing, ACM, New York, 1990, pp. 73–83.

[13] M. Blum and S. Micali, *How to generate cryptographically strong sequences of pseudo-random bits*, SIAM Journal on Computing, 13 (1984), pp. 850–864.

[14] J. Cai, *With probability one, a random oracle separates PSPACE from the polynomial-time hierarchy*, Journal of Computer and System Sciences, 38 (1989), pp. 68–85.

[15] J. Feigenbaum, S. Kannan, and N. Nisan, *Lower bounds on random-self-reducibility*, in Proceedings of the 5th Structure in Complexity Theory Conference, IEEE Computer Society, Los Alamitos, 1990, pp. 100–109.

[16] S. Fenner, L. Fortnow, and S. Kurtz, *Gap-definable counting classes*, in Proceedings of the 6th Structure in Complexity Theory Conference, IEEE Computer Society, Los Alamitos, 1991, pp. 30–42.

[17] S. Goldwasser and S. Micali, *Probabilistic encryption*, Journal of Computer and System Sciences, 28 (1984), pp. 270–299.

[18] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof-systems*, SIAM Journal on Computing, 18 (1989), pp. 186–208.

[19] S. Goldwasser and M. Sipser, *Private coins versus public coins in interactive proof systems.*, in Randomness and Computation, ed.: S. Micali, vol. 5 of Advances in Computing Research, JAI Press, Greenwich, 1989, pp. 73–90.

[20] U. Hertrampf, *Relations among MOD-classes*, Theoretical Computer Science, 74 (1990), pp. 325–328.

[21] R. Lipton, *New directions in testing*, in Distributed Computing and Cryptography, eds.: J. Feigenbaum and M. Merritt, vol. 2 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, 1991, pp. 191 – 202.

[22] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, *Algebraic methods for interactive proof systems*, Journal of the ACM, 39 (1992), pp. 859–868.

[23] C. Papadimitriou and S. Zachos, *Two remarks on the power of counting*, in Proceedings of the 6th GI Conference on Theoretical Computer Science, vol. 145 of Lecture Notes in Computer Science, Springer, Berlin, 1983, pp. 269–276.

[24] M. Rabin, *Probabilistic algorithms in finite fields*, SIAM Journal on Computing, 9 (1980), pp. 273–280.

[25] A. Shamir, *IP = PSPACE*, Journal of the ACM, 39 (1992), pp. 869–877.

[26] J. Spencer, *Ten Lectures on the Probabilistic Method*, vol. 52 of CBMS, SIAM, Philadelphia, 1987.

[27] S. Toda, *PP is as hard as the polynomial-time hierarchy*, SIAM Journal on Computing, 20 (1991), pp. 865–877.

[28] M. Tompa and H. Woll, *Random self-reducibility and zero-knowledge interactive proofs of possession of information*, in Proceedings of the 28th Symposium on Foundations of Computer Science, IEEE Computer Society, Los Alamitos, 1987, pp. 472–482.

[29] L. Valiant, *The complexity of computing the permanent*, Theoretical Computer Science, 8 (1979), pp. 189–201.

[30] C. Yap, *Some consequences of nonuniform conditions on uniform classes*, Theoretical Computer Science, 26 (1983), pp. 287–300.