[12] A. Shamir. $IP = PSPACE$, J. ACM, 39 (1992), pp. 869–877.

6 Acknowledgements

We thank Sampath Kannan, Noam Nisan, and Ronitt Rubinfeld for early discussions about the relationship between self-correction and random-self-reducibility. The third author wishes to thank his advisor Alan Selman for his encouragement and advice, and Jin-yi Cai and Alan Selman for useful discussions.

References

[1] L. Babai, L. Fortnow, and C. Lund. *Non-Deterministic Exponential Time has Two-Prover Interactive Proofs*, Computational Complexity, 1 (1991), pp. 3–40.

[2] D. Beaver and J. Feigenbaum. *Hiding Instances in Multioracle Queries*, in Proc. 7th Annual Symposium On Theoretical Aspects Of Computer Science, Lecture Notes in Computer Science, vol. 415, Springer, Berlin, 1990, pp. 37–48.

[3] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. *Security with Low Communication Overhead*, in Advances in Cryptology – Crypto '90, Lecture Notes in Computer Science, vol. 537, Springer, Berlin, 1991, pp. 62–76.

[4] M. Blum, M. Luby, and R. Rubinfeld. *Self-testing/correcting, with Applications to Numerical Problems*, J. Comput. System Sci., 47 (1993), pp. 549–595.

[5] R. Book. *Tally Languages and Complexity classes*, Information and Control, 26 (1974), pp. 186–193.

[6] J. Feigenbaum and L. Fortnow. *Random-Self-Reducibility of Complete Sets*, SIAM J. Comput., 22 (1993), pp. 994–1005.

[7] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson. *Self-testing/correcting for Polynomials and for Approximate Functions*, in Proc. 23rd Symposium on the Theory of Computing, ACM, New York, 1991, pp. 32–42.

[8] P. Gemmell and M. Sudan. *Highly Resilient Correctors for Polynomials*, Inf. Proc. Letters, 43 (1992), pp. 169–174.

[9] R. Lipton. *New Directions in Testing*, in Distributed Computing and Cryptography, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 2, American Mathematical Society, Providence, 1991, pp. 191–202.

[10] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. *Algebraic Methods for Interactive Proof Systems*, J. ACM, 39 (1992), pp. 859–868.

[11] R. Rubinfeld and M. Sudan. *Testing Polynomial Functions Efficiently and over Rational Domains*, Proc. 3rd Symposium on Discrete Algorithms, ACM/SIAM, New York/Philadelphia, 1992, pp. 23–32.

That (σ'', ϕ'') satisfies requirement 1 (the correctness property) of Definition 2.1 follows directly from the correctness of (σ', ϕ'). That it satisfies requirement 2 (the local randomness property) follows from Equation (4): Each query z_i, $1 \leq i \leq (k' + k'')(n)$, is distributed according to $\mathcal{D}_n$, and thus its distribution does not depend on the specific input $x \in \{0, 1\}^n$.

We now show how to obtain the necessary ingredients k'' and Pr''. Let b be a positive integer such that, for all z, $Pr'(x, z)$ and $P_{D_n}(z)$ are integral multiples of 2^{-b}. Such a b can always be found in our model of computation, where the source of randomness is simply an unbiased coin. Rearranging Equation (4) yields

$$Pr''(x, z) = \frac{(k' + k'')(n) P_{D_n}(z) - k'(n) Pr'(x, z)}{k''(n)}. \tag{5}$$

Set $k''(n)$ to be any integer of the form 2^c that is large enough to make $Pr''(x, z)$ positive for all z but small enough so that $2^c = poly(n)$. That such a 2^c exists is guaranteed by the definition of superfluous (see Equation (3)) and the fact that σ' only asks queries that are *not* superfluous.

From Equation (5), we now see that $Pr''(x, z)$ is an integral multiple of $2^{-(b+c)}$. Let

$$g_x(z) = \sum_{w<z} Pr''(x, w) 2^{b+c}.$$

Observe that g_x is a #P function and hence by assumption in FP. In order to generate random strings in such a way that each z is produced with probability $Pr''(x, z)$, simply choose a string v by flipping $b + c$ coins and then use binary search to output the largest z for which $g_x(z) < v$.

This completes the proof of the theorem. □

5 Open Questions

Theorem 3.1 provides some evidence that the notions of self-correctability and random-self-reducibility are different. However, observe that the proof of this theorem does *not* give a self-corrector whose ensemble is P-sampleable; moreover, the fact that the ensemble is not P-sampleable is critical in showing that χ_L is not nonadaptively rsr. Is there a hypothesis that implies the existence of a function that is nonadaptively self-correctable with respect to a P-sampleable ensemble but not nonadaptively rsr? Are there relativized worlds in which these two notions can be separated?

It would also be interesting to know whether there is a result analogous to Theorem 4.1 in which the self-corrector and the rsr are adaptive.

the probability that $z = \sigma(1, x, r)$, where r, as usual, is chosen uniformly at random from $\{0,1\}^{w(n)}$. (We use $\sigma(1, x, r)$ for concreteness. Because all the random variables $\sigma(i, x, r)$ are identically distributed, $Pr(x, z)$ could have been defined in terms of any of them and would take on exactly the same values.) Then a query z is superfluous if $Pr(x, z)$ is much bigger than $\mathrm{Prob}_{\mathcal{D}_n}(z)$. Specifically, if

$$Pr(x, z) \geq \frac{1}{\epsilon^2(n)} \cdot k(n) \cdot \mathrm{Prob}_{\mathcal{D}_n}(z), \tag{3}$$

then we say that z is superfluous with respect to x.

The next fact follows straightforwardly from the definitions.

Fact 4.2 For all n and all $x \in \{0,1\}^n$, the probability that a random z, drawn according to $\mathcal{D}_n$, is superfluous with respect to x is at most $\epsilon^2(n)$.

Consider a self-corrector (σ', ϕ') that works as follows. On input x, σ' first computes $z_1, \ldots, z_{k(n)}$, a set of random queries produced by σ. For each i, σ' decides whether z_i is superfluous with respect to x; note that assumptions (B) and (C) above make this doable in polynomial time. If z_i is superfluous, then the corrector assumes that $P(z_i) = 0$; otherwise, it queries the program to find out $P(z_i)$. If $k'(n)$ denotes the maximum number of queries to P made by σ', then k' is clearly bounded above by k. The function ϕ' then computes $f(x)$ exactly as ϕ would have. Once again, the following fact is straightforward.

Fact 4.3 (σ', ϕ') is a nonadaptive $(\epsilon - \epsilon^2)$-self-corrector for f.

Let $Pr'(x, z)$ be the probability that $\sigma'(1, x, r) = z$. Note that $Pr'(x, z) = Pr(x, z)$ if z is not superfluous and is equal to 0 if z is superfluous.

Our nonadaptive rsr (σ'', ϕ'') has the following structure. On input x, it produces $k'(n)$ of its queries by calling σ'. It produces an additional $k''(n)$ queries according to a P-sampleable ensemble described below. It reconstructs $f(x)$ by applying ϕ' to the answers to the queries produced by σ', i.e., it ignores the answers to the other $k''(n)$ queries. The reason for asking these additional $k''(n)$ queries is to make its query-distribution dependent only on n, rather than on the specific input x.

More precisely, we seek a polynomially bounded function $k''(n)$ and a P-sampleable ensemble $\{Pr''(x, z)\}_x$ that satisfy the following equation for all z.

$$\frac{k'(n)Pr'(x, z) + k''(n)Pr''(x, z)}{(k' + k'')(n)} = P_{D_n}(z) \tag{4}$$

The nonadaptive rsr (σ'', ϕ'') is then fully specified as follows. On input x, first run σ' to produce $z_1, \ldots, z_{k'(n)}$. Then do $k''(n)$ independent samplings of Pr'' to produce $z_{k'(n)+1}, \ldots, z_{(k'+k'')(n)}$. Randomly permute the queries $z_1, \ldots, z_{(k'+k'')(n)}$ and submit them to the oracle. Apply the appropriate inverse permutation to extract the answers $f(z_1), \ldots, f(z_{k'(n)})$. Use these answers and the function ϕ' to compute $f(x)$.

Proof of Claim Suppose R_i is marked fulfilled at stage n. The claim is obvious if R_i is marked as fulfilled because the distribution of queries to f is not identical for all $x \in \Sigma^n$.

Now, assume that the distribution of queries to L by M_i is identical for all inputs $x \in \Sigma^n$. Recall that, if R_i is marked fulfilled, then the probability that u_n is queried is less than 1/4. Suppose it holds that $\|S_0\| > \|S_1\|$. This implies that $u_n \in L$. We claim that the probability that M_i accepts u_n is less than 3/4. This follows from the facts that $\|\mathcal{S}_w\|$ is less than 1/4 of the total number of random strings and that a majority (that is, $\|\mathcal{S}_0\|$) of the remaining random strings yield $M_i(u_n) = 0$. Thus, $\|\mathcal{S}_w\| + \|\mathcal{S}_1\|$ is less than 3/4 of the total random strings. The proof for the case $\|\mathcal{S}_0\| \leq \|\mathcal{S}_1\|$ is identical. □

This completes the proof of the theorem. □

4 Collapse

In this section, we relate the question of self-correctability versus random-self-reducibility to the question of #P versus FP.

Theorem 4.1 *Suppose that*

(A) f is nonadaptively self-correctable with respect to $\mathcal{D}$,

(B) $\mathcal{D}$ is P-sampleable, and

(C) #P $\subseteq$ FP.

Then f is nonadaptively random-self-reducible.

Proof We first note that (B) and (C) together imply that $\mathrm{Prob}_{\mathcal{D}}(x)$ is a polynomial-time computable function. Let (ϕ, σ) be an ϵ-self-corrector for f with respect to $\mathcal{D}$, where $\epsilon(n) = 1/poly(n)$, and assume without loss of generality that this corrector produces a wrong answer with probability at most 2^{-n}. Let the number of random queries produced by σ be $k(n)$. We will construct a nonadaptive rsr for f.

Recall that the key difference between a nonadaptive rsr and a nonadaptive self-corrector is that the definition of a self-corrector allows its function σ to produce random queries whose distribution depends on the input x. The definition of a nonadaptive rsr, on the other hand, explicitly disallows this, insisting that the distribution depend only on the length n of x. Note, however, that Definition 2.2 does refer to a distribution, namely $\mathcal{D}_n$, that depends only on n. We will use $\mathcal{D}_n$ to define a distribution that can be used in a nonadaptive rsr.

A central notion in our construction is that of a query that is *superfluous* with respect to a given input x. Intuitively, z is superfluous if the answer $P(z)$ that a self-corrector gets by querying the program about z cannot be very useful in computing $f(x)$. Let $Pr(x, z)$ denote

$c > 2$, and $i \leq n$, it follows that, for all n,

$$p_i(2^{2^{p(n)}}) = 2^{i2^{p(n)}} < 2^{2^{p(n+1)}}.$$

Thus, for sufficiently large n, $M_i(1^n)$ cannot query any string of good length m such that $m > n$. This implies that, if a simulation of M_i does not query u_n, its output does not depend on the membership of u_m in L, for any $m \geq n$. We partition the set $\Sigma^{w_i(n)}$ into the following three sets.

$$\begin{aligned} \mathcal{S}_w &= \{r \mid M_i(u_n, r) \text{ queries } u_n\ \} \\ \mathcal{S}_0 &= \{r \mid M_i(u_n, r) \text{ does not query } u_n \text{ and outputs } 0\} \\ \mathcal{S}_1 &= \{r \mid M_i(u_n, r) \text{ does not query } u_n \text{ and outputs } 1\} \end{aligned}$$

If $\|\mathcal{S}_0\| > \|\mathcal{S}_1\|$, then leave u_n in L, else add u_n to $\overline{L}$. Finally, mark requirement R_i fulfilled and go to the next stage. This completes the construction of L.

The proof that χ_L is not nonadaptively rsr is implied by the following two claims.

Claim 3.3 *For all i, there is a stage ℓ such that R_i is fulfilled at stage ℓ.*

Proof of Claim We prove this by contradiction. Suppose that at least one requirement remains unfulfilled at the end of the construction. Note that this means that there must be a unique, smallest i such that R_i remains unfulfilled. Let N be the smallest stage at which we attempted to fulfill this R_i. Recall that R_i can remain unfulfilled only if, for all good numbers m such that $1^m \in T$ and $m > N$

$$Pr[M_i(1^m) \text{ queries } u_m\ ] \geq 1/4.$$

We show that the above equation contradicts the hypothesis that T is in $\mathrm{UP} - \mathrm{R}$ by giving an RP algorithm for T.

On input 1^n, if $n < N$, then use a finite look-up table to determine whether $1^n \in T$. If $n \geq N$, then simulate the query-generation part of $M_i(1^n)$ (i.e., the calls to the function σ of Definition 2.1) n times. If u_n belongs to any of these n sets of queries, then accept 1^n, else reject.

If $1^n \in T$, then the above algorithm accepts it with probability at least $\geq 1 - (3/4)^n$, which for $n > 6$ is greater than $3/4$. If $1^n \notin T$, then the witness u_n does not exist and hence is never generated in any of the n simulations of $M_i(1^n)$; this means that the algorithm never accepts an input 1^n that is not in T. Thus, this is an RP algorithm for T. □

Claim 3.4 *For all i, if R_i is marked fulfilled using the above procedure, then M_i is not a nonadaptive rsr for L.*

Claim 3.2 *If L is a subset of $\mathcal{W}$, then, for all $\epsilon < 1$, χ_L is ϵ-self-correctable with respect to $\mathcal{D}$.*

Proof of Claim It follows directly from the definition of $\mathcal{D}$ that, for any $\epsilon < 1$, any program P that computes χ_L with error probability at most ϵ with respect to $\mathcal{D}$ cannot commit an error on any string in $\mathcal{W}$. Furthermore, membership in $\mathcal{W}$ can be tested in polynomial time. Thus, a self-corrector M for χ_L works as follows on input x: If $x \in \mathcal{W}$, then M outputs $P(x)$; else M outputs 0. □

By Claim 3.2, it suffices to construct a language L that is a subset of $\mathcal{W}$ but is not nonadaptively rsr. We give a recursive procedure that constructs such an L.

Suppose $\{M_i\}$ is an enumeration of all probabilistic polynomial-time nonadaptive oracle Turing machines in which the running time of M_i is $p_i(n) = n^i$. We assume that each probabilistic Turing machine M_i is implemented as a deterministic machine that takes two arguments as input, namely, an input string x and a random string r such that $|r| = w_i(|x|)$ for some polynomial w_i. The language L is constructed such that for all $i \in N$, the following set of requirements $\{R_i\}$ is fulfilled:

(*) R_i : M_i is not a nonadaptive rsr for χ_L

We fulfill these requirements in stages. Initially, let $\overline{L}$ be $\overline{\mathcal{W}}$. At stage n, we decide the membership in L of the string u_n, if such a string exists. If $1^n \in T$, then we attempt to fulfill a requirement at stage n; if we succeed, then we mark the requirement *fulfilled*, else, we mark it *unfulfilled*. We assume that all the requirements are initially marked unfulfilled.

The main idea of the construction is as follows. The language L will be a subset of $\mathcal{W}$ and hence nonadaptively self-correctable. If all of the requirements are fulfilled during the construction, L will obviously not be nonadaptively rsr. We show that all the requirements will in fact be fulfilled by showing that, otherwise, T is in R, which contradicts the hypothesis of the theorem.

At stage n, if $1^n \notin T$, then do nothing and go to the next stage. Else, if $1^n \in T$, then let i be the smallest integer such that requirement R_i is as yet unfulfilled.

If the distribution of the queries made by M_i to the oracle is not identical for all strings in Σ^n (that is, there are two strings y and z in Σ^n and a $q \in \Sigma^*$ such that the probability of querying q on inputs y and z is different), then add u_n to $\overline{L}$, mark R_i as fulfilled, and go to the next stage.

Next assume that the queries to the oracle by M_i on all inputs $x \in \Sigma^n$ have identical distributions. One of the following equations must hold.

$$Pr_r[M_i(1^n, r) \text{ queries } u_n\,] \;\geq\; 1/4, \text{ or} \tag{1}$$

$$Pr_r[M_i(1^n, r) \text{ queries } u_n\,] \;<\; 1/4. \tag{2}$$

If Equation (1) holds, then add u_n to $\overline{L}$ and go to the next stage. If Equation (2) holds, then, for all strings r in $\Sigma^{w_i(n)}$, simulate $M_i(u_n, r)$. Because $p(n) = n^c$, for some integer

As observed by Szegedy and noted in [6], we may assume that all of the random queries $\sigma(i, x, r)$, $1 \leq i \leq k(n)$, produced by a nonadaptive rsr or a nonadaptive self-corrector are identically distributed. If they are not, then define a new σ that applies a random permutation to the queries before it asks them and applies the inverse permutation to the answers before it delivers them to ϕ. Each of the queries produced by this new σ is distributed according to the average of the distributions of the queries of the original σ.

Finally, note that, although the explicit requirement given in Definitions 2.1 and 2.2 is that the probability that a random-self-reduction or self-corrector produces a wrong answer is at most 1/4, we may assume without loss of generality that this probability is at most 2^{-n}, by the standard application of Chernoff bounds, provided we are willing to tolerate a polynomial-factor increase in $k(n)$, the number of queries made by σ.

3 Separation

In this section, we show that there is a reasonable hypothesis that implies that there exists a function f that is nonadaptively self-correctable for some ensemble $\mathcal{D}$ but not nonadaptively random-self-reducible.

Let UEEEXP denote the class of languages that are accepted by unambiguous nondeterministic Turing machines that run in nondeterministic time $2^{2^{2^{p(n)}}}$ for some polynomial p; similarly, let REEEXP denote the class of languages accepted by probabilistic Turing machines with one-sided error in time $2^{2^{2^{p(n)}}}$ for some polynomial p. For any language L, let χ_L denote the characteristic function of L.

Theorem 3.1 *If* UEEEXP $\not\subseteq$ REEEXP, *then there exists a language L and an ensemble $\mathcal{D} = \{\mathcal{D}_n\}_{n\geq 1}$ such that, for all $\epsilon < 1$, χ_L is nonadaptively ϵ-self-correctable with respect to $\mathcal{D}$ but* not *nonadaptively random-self-reducible.*

Proof By standard padding arguments [5], it follows from the hypothesis that there exists a language $T \in$ UP $-$ R such that, for some polynomial p, $T \subseteq \{1^{2^{2^{p(n)}}} \mid n \in N\}$. Without loss of generality, assume that p is a monomial n^c, for some integer $c > 2$; this ensures that there are sufficiently large gaps between consecutive elements of T. We say that a natural number n is *good* if there exists an integer j such that $n = 2^{2^{p(j)}}$. For all n such that $1^n \in T$, let u_n be the unique witness of this fact. Assume, without loss of generality, that $|u_n| = n$, for all n such that $1^n \in T$. Finally, let $\mathcal{W}$ be the set of all witnesses u_n.

For each $n \geq 1$, define a distribution $\mathcal{D}_n$ on Σ^n as follows: If $1^n \in T$, then for each $x \in \Sigma^n$:

$$Pr_{\mathcal{D}_n}(x) = 1, \text{ if } x \in \mathcal{W}$$
$$Pr_{\mathcal{D}_n}(x) = 0, \text{ otherwise}$$

If $1^n \notin T$, then for each $x \in \Sigma^n$:

$$Pr_{\mathcal{D}_n}(x) = \frac{1}{2^n}.$$

Let $\mathcal{D}$ be the ensemble $\{\mathcal{D}_n\}_{n\geq 1}$. All subsets of $\mathcal{W}$ have the following useful property.

2 Definitions

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 random-self-reduction or a self-corrector, denoted $k(n)$, is also a polynomially bounded function of n.

The following definition of nonadaptive random-self-reducibility was first stated formally in [6]. It is a special case of the more general notion of "locally random reducibility" first formalized in [3].

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 $x \in \{0,1\}^n$,

$$f(x) = \phi(x, r, f(\sigma(1,x,r)), \ldots, f(\sigma(k(n),x,r)))$$

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

2. For all n and all i, $1 \leq i \leq k(n)$, if $|x_1| = |x_2| = n$ and r is chosen uniformly from $\{0,1\}^{w(n)}$, then the random variables $\sigma(i, x_1, r)$ and $\sigma(i, x_2, r)$ are identically distributed.

In this paper, we say simply that "f is nonadaptively rsr" if it is nonadaptively $poly(n)$-random-self-reducible and the precise polynomially bounded number of queries does not matter.

In the following definition of self-correction, based on the one in [4], $\mathcal{D}$ denotes an ensemble $\{\mathcal{D}_n\}_{n\geq 1}$ of probability distributions. The ensemble is *P-sampleable* if there is a probabilistic polynomial-time algorithm that, on input 1^n, outputs each string z with probability exactly $Prob_{\mathcal{D}_n}(z)$. If P is a program that purports to compute f, then $error(P, f, \mathcal{D}_n)$ is the probability that $P(z) \neq f(z)$, when z is chosen randomly according to $\mathcal{D}_n$. If $error(P, f, \mathcal{D}_n) \leq \epsilon(n)$, for all $n \geq 1$, we say that $error(P, f, \mathcal{D}) \leq \epsilon$.

Definition 2.2 A nonadaptive **$\epsilon(n)$-self-corrector for f with respect to $\mathcal{D}$** consists of a pair of polynomial-time computable functions σ and ϕ with the following property: If $error(P, f, \mathcal{D}) \leq \epsilon$, then, for all x, $f(x) = \phi(x, r, P(\sigma(1,x,r)), \ldots, P(\sigma(k(n),x,r)))$ for at least $3/4$ of all r's in $\{0,1\}^{w(n)}$.

Adaptive versions of both random-self-reductions and self-correctors can be defined; see [4, 6] for details.

We say that "f is nonadaptively self-correctable with respect to $\mathcal{D}$" if it has a nonadaptive $\epsilon(n)$-self-corrector with respect to $\mathcal{D}$ for some function $\epsilon(n) = 1/poly(n)$. Blum, Luby, and Rubinfeld [4] give a simple proof that if f is nonadaptively rsr, then it is nonadaptively self-correctable.

In order to recover $g_n(x)$ from $g_n(y_1)$, …, $g_n(y_{n+1})$, the reduction simply interpolates the $n+1$ values $(\alpha_1, G(\alpha_1)), \ldots, (\alpha_{n+1}, G(\alpha_{n+1}))$ to obtain the polynomial G and outputs the constant term.

Recall that there are multilinear polynomial families of arbitrarily high computational complexity. Thus random-self-reducibility and self-correction are not properties that are limited to functions in a particular complexity class.

This example of a random-self-reduction was first developed in [2, 9] and subsequently played a crucial role in the characterization of the language-recognition power of interactive proof systems [1, 10, 12]. As observed by Gemmell *et al.* [7], it also provides a good example of how a random-self-reduction can be interpreted as a self-corrector. Suppose that P is a program that purports to compute g and that, for every n, the fraction of inputs in K_n^n on which P errs is known to be at most $1/5n$. Then the standard random-self-reduction for multilinear polynomials serves as a self-corrector for P, and its output is correct with probability at least 3/4, on all inputs. Gemmell *et al.* [7] show how to construct a more powerful corrector, i.e., one that tolerates a constant fraction of errors; further improvements were subsequently made by, e.g., Rubinfeld-Sudan [11] and Gemmell-Sudan [8].

The multilinear-polynomial example illustrates a general principle that was observed in the original paper by Blum, Luby, and Rubinfeld [4]: If a function is random-self-reducible, then it is self-correctable. This observation immediately raises the question of whether the two notions are formally equivalent, i.e., whether any self-correctable function is also random-self-reducible. Indeed the known self-correctors all use some form of random-self-reducibility.

Note that the local randomness property imposed on all instances produced by a random-self-reduction is more stringent than the requirements imposed on a self-corrector. Theoretically, the distribution on queries produced by a self-corrector could be highly dependent on the particular input instance x. However, none of the previously known self-correctors used this ability. Our first result demonstrates the existence of self-correctors that use this ability.

> **Theorem**: If UEEEXP $\not\subseteq$ REEEXP, then there exists a function f that is nonadaptively self-correctable but *not* nonadaptively random-self-reducible.

The self-corrector for the function f in this theorem only queries the program about the input string; so its queries are dependent on the input in a very strong sense. Also, this corrector is very powerful; it can tolerate any constant fraction of errors. However, the function f is not nonadaptively random-self-reducible. This theorem is proved in Section 3.

How hard is it to construct nonadaptively self-correctable functions that are not nonadaptively random-self-reducible? Our next result addresses this question.

> **Theorem**: If #P $\subseteq$ FP, then every function that is nonadaptively self-correctable with respect to a P-sampleable ensemble is also nonadaptively random-self-reducible.

Thus, constructing functions that are nonadaptively self-correctable with respect to P-sampleable ensembles but not nonadaptively random-self-reducible is as hard as proving #P $\not\subseteq$ FP. Although the assumption that #P $\subseteq$ FP is generally believed to be false, it is not likely to be disproven any time soon. We prove our second result in Section 4.

1 Introduction

The complexity-theoretic notions of program self-correction and random-self-reducibility have been carefully studied and are known to be closely related. However, whether the two notions are formally equivalent remains open. In this note, we continue the study in an attempt to shed light on this open question.

Blum, Luby, and Rubinfeld [4] defined program self-correction in order to address the following question. Let P be a program for which one can determine that the number of inputs on which P errs, while not necessarily zero, is limited. Is it possible to write an auxiliary program C that corrects P's errors with high probability? More precisely, on any input, C should produce the correct answer with high probability, and C may call the (potentially faulty) program P several times in the course of its computation.

Random-self-reducible functions have long played an important role in many areas of computer science, e.g., in cryptographic protocols and in average-case complexity. (See [6] for a thorough discussion of this role.) If f is random-self-reducible, then 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. The sense in which each y_i must be random is very strong: If given instances x_1 and x_2 are of the same length, then the distribution induced by the random-self-reduction on random instance y_i must be the same, whether the reduction starts with input x_1 or x_2. This requirement is sometimes called the *local randomness property*, and it allows one to show, for example, that random-self-reducible functions are, in a very precise sense, as hard on average as they are in the worst case.

Multilinear polynomials over finite fields provide well-known examples of random-self-reducible functions. Let $K = \{K_n\}_{n\geq 1}$ be a sequence of finite fields such that $|K_n| > n$, and let $g = \{g_n\}_{n\geq 1}$ be a sequence of multilinear polynomials over these fields such that $g_n \in K_n[X_1, \ldots, X_n]$. Because g_n is multilinear, it is of total degree at most n.

Let $x = (x_1, \ldots, x_n)$ be an element of K_n^n for which we would like to determine $g_n(x)$. The reduction proceeds as follows. Let α_1, …, α_{n+1} be distinct elements of K_n. Choose coefficients c_1, …, c_n independently and uniformly at random from K_n. Let $y_i = (c_1\alpha_i + x_1, \ldots, c_n\alpha_i + x_n)$, for $1 \leq i \leq n+1$. Because the c_j's were chosen randomly and independently, the induced distribution on each y_i is uniform over K_n^n and independent of the original input x (although y_i and $y_{i'}$ are dependent on each other). Consider the one-variable polynomial $G \in K_n[Z]$ defined by the equation

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

Note that G is of degree at most n and hence determined by its values on $n+1$ distinct values of Z. Furthermore, G satisfies

$$G(\alpha_i) = g_n(y_i),$$

and in particular it satisfies

$$G(0) = g_n(x).$$

ABSTRACT

We examine the relationship between two types of probabilistic self-reductions that play crucial roles in the theory of interactive proof systems, program-checking, and program-testing: *self-correctors* and *random-self-reductions*. It is well known [4] that if a function is random-self-reducible, then it is also self-correctable; indeed all known self-correctors use some form of random-self-reducibility. However, whether self-correctability implies random-self-reducibility, i.e., whether the two properties are equivalent, remains an important open question.

We show that:

(i) If UEEEXP $\not\subseteq$ REEEXP, then there there exists a function f that is nonadaptively self-correctable but *not* nonadaptively random-self-reducible.

(ii) If #P $\subseteq$ FP, then every function that is nonadaptively self-correctable with respect to a P-sampleable ensemble is also nonadaptively random-self-reducible.

DIMACS Technical Report 94-45
August 1994

Two Remarks on Self-Correctability versus Random-Self-Reducibility

by

Joan Feigenbaum[1]
AT&T Bell Laboratories
Room 2C-473
600 Mountain Avenue
Murray Hill, NJ 07974
`jf@research.att.com`

Lance Fortnow[2]
Computer Science Department
University of Chicago
1100 E. 58th Street
Chicago, IL 60637 USA
`fortnow@cs.uchicago.edu`

Ashish V. Naik[3]
Department of Computer Science
State University of New York
Buffalo, NY 14260 USA
`avnaik@cs.buffalo.edu`

[1] Permanent Member of DIMACS

[2] Supported in part by NSF grant CCR 92-53582.

[3] Supported in part by NSF grant CCR 90-02292.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.