数理解析研究所講究録
第 871 巻 1994 年 138-144

On Checkers, Self-Testers, and Self-Debuggers

東京工業大学総合理工学研究科 森 啓悦 (Hiroyoshi Mori)
東京工業大学総合理工学研究科 伊東 利哉 (Toshiya Itoh)

1 Introduction

1.1 Background and Motivation

When a programmer writes a program P_f that computes a function f, one of the main difficulties is to mathematically prove that P_f is correct. Blum and Kannan [BK] first attempted to overcome this problem and introduced a notion of "program checking" as an application of interactive proofs [GMR], [BM]. Informally, a function f is said to have a (program) checker C_f if (1) when P_f is correct, C_f making calls to P_f outputs with high probability "**correct**" on any input $x \in \{0,1\}^*$; and (2) when P_f is incorrect, C_f making calls to P_f outputs with high probability "**incorrect**" on any input $x \in \{0,1\}^*$ such that $P_f(x) \neq f(x)$. Here we note that the output of a checker C_f only verifies the correctness of the output of any program P_f on specified input $x \in \{0,1\}^*$ but does not guarantees the correctness of the program P_f being checked. Then as an extension of checkers, Blum, Luby, and Rubinfeld [BLR] introduced a notion of "self-tester/corrector," and "self-tester/corrector pair," which is a powerful tool in a practical setting. Informally, a function f is said to have a self-tester ST_f if (1) when P_f is tolerably faulty, ST_f making calls to P_f outputs "**pass**" with high probability; and (2) when P_f is too faulty, ST_f making calls to P_f outputs "**fail**" with high probability, and a function f is said to have a self-corrector SC_f if when P_f is not too faulty, SC_f making calls to P_f outputs with high probability a correct answer for every input to f. In addition, Blum, Luby, and Rubinfeld [BLR] introduced a notion of "self-tester/corrector pair," which is a powerful tool in a practical setting.

Of course, we may expect that every function has a checker, a self-tester, and a self-corrector, however, it is not the case. Indeed, there exists a function $f : \{0,1\}^* \mapsto \{0,1\}$ that does not have a checker C_f under some complexity-theoretic assumption [Y], [BF], [BG]. As a relationship among checkers, self-testers, and self-correctors, Blum, Luby, and Rubinfeld [BLR] showed that if a function f has a checker C_f, then f has a self-tester ST_f (see Theorem 2.8). This implies that the task of self-testers is not harder than that of checkers. On the contrary, it is not known whether or not every function that has a checker has a self-corrector. Then

- **Question 1:** Does every function f that has a checker have a self-corrector?

Intuitively, the task of self-correctors seems to be much harder than that of checkers. In section 3, we show that it is indeed the case assuming the existence of oneway permutations.

Blum, Luby, and Rubinfeld [BLR] also showed that if a function f has a self-tester/corrector pair $\langle \overline{ST}_f, \overline{SC}_f \rangle$, then f has a checker C_f. It follows from the result above (see Theorem 2.8) that if a function f has a self-tester/corrector pair $\langle \overline{ST}_f, \overline{SC}_f \rangle$, then f has a self-tester ST_f. It should be noted that the resulting self-tester ST_f for f could be very reliable but it is somewhat inflexible for a practical purpose, i.e., ST_f only passes completely correct programs P_f for f but does not pass almost correct programs P_f for f. Then

- **Question 2:** How can we make self-testers for a function f reliable and flexible?

To investigate this problem, we introduce in section 4 a novel notion of "self-debuggers" for f and then present in subsection 5.1 a way to design a reliable and flexible self-tester for f from any self-tester/corrector pair for f.

1.2 Results

In this paper, we first present a negative solution to Question 1, i.e., if oneway permutations exist, then there exists a function $f : \{0,1\}^* \mapsto \{0,1\}$ that has a checker but does not have a self-corrector (see Theorem 3.1). To demonstrate a positive solution to Question 2, we introduce in section 4 a novel notion of "self-debuggers" and then show that if a function f has a self-tester/corrector pair $\langle \overline{ST}_f, \overline{SC}_f \rangle$, then f has a self-debugger SD_f (see Theorem 4.2). As the applications of self-debuggers, we show in subsection 5.1 that for any $0 < \varepsilon_1' < \varepsilon_2' \leq \varepsilon' < 1$, if $\overline{ST}_f$ is an $(\varepsilon_1', \varepsilon_2')$-self-tester for f and $\overline{SC}_f$ is an ε'-self-corrector for f, then for any $\varepsilon_1 < \varepsilon_2$ such that $\varepsilon_1 \leq \varepsilon'$, there exists an $(\varepsilon_1, \varepsilon_2)$-self-tester ST_f for f (see Theorem 5.1) and in subsection 5.2 that for any $0 < \varepsilon_1' < \varepsilon_2' \leq \varepsilon' < 1$, if $\overline{ST}_f$ is an $(\varepsilon_1', \varepsilon_2')$-self-tester for f and $\overline{SC}_f$ is an ε'-self-corrector for f, then for any constant $\delta > 0$ such that $\delta \leq \varepsilon'$, there exists an (ε', δ)-self-debugger SD_f for f (see Theorem 5.2). The result of Theorem 3.1 captures our intuition that the task of self-correctors seems to be much harder than that of checkers and the results of Theorems 5.1 and 5.2 provides us a powerful tool in a practical setting.

2 Preliminaries

In this section, we first present definitions and terminologies necessary to the subsequent technical discussions and then overview the results known so far.

2.1 Definitions and Terminologies

Let $f : \{0,1\}^* \mapsto \{0,1\}^*$ be a function and let P_f be a program that purports to compute f. In this paper, any program P_f is assumed to be *static*, i.e., $P_f(x)$ is completely determined by the current input $x \in \{0,1\}^*$ and does not depend on any input previously asked of the program P_f. We use $P_f(x)$ to denote the output of a program P_f on input $x \in \{0,1\}^*$. A program P_f for f is said to be *correct* if $P_f(x) = f(x)$ for every $x \in \{0,1\}^*$ and a program P_f for f is said to be *incorrect* if there exists $w \in \{0,1\}^*$ such that $P_f(w) \neq f(w)$.

Definition 2.1 [BLR]: *A (probabilistic) program M is said to be an* oracle program *if it is allowed to make calls to another program that is specified at its run time. We use M^A to denote an oracle program M that makes calls to another program A.*

Definition 2.2 [BK]: *A probabilistic polynomial time oracle program C_f is said to be a* checker *for a function $f : \{0,1\}^* \mapsto \{0,1\}^*$ if for every program P_f that purports to compute f, C_f on input $x \in \{0,1\}^*$ satisfies the following conditions:*

(1) *If P_f is* correct, *then* $\Pr\{C_f^{P_f}(x) = \text{"correct"}\} \geq 2/3$ *for any $x \in \{0,1\}^*$;*

(2) *For any $x \in \{0,1\}^*$ such that $P_f(x) \neq f(x)$,* $\Pr\{C_f^{P_f}(x) = \text{"incorrect"}\} \geq 2/3$,

where the probabilities are taken over all possible coin tosses of C_f.

Note that in Definition 2.2, the output of $C_f^{P_f}$ on any input $x \in \{0,1\}^*$ such that $P_f(x) = f(x)$ is not specified when P_f is incorrect.

We say that a function f is *checkable* (or f has a checker) if there exists a checker C_f for f and also that a language L is *checkable* (or L has a checker) if there exists a checker C_L for its characteristic function λ_L. A function f (or a language L) is uncheckable if it is not checkable.

Let N be a set of positive integers and let $I \subseteq \{0,1\}^*$. Let $I_1, I_2, \cdots \subseteq I$ be a sequence of subsets of I that satisfies $I_1 \cup I_2 \cup \cdots = I$. Note that each $n \in N$ indicates the "size" of each $x \in I_n$. We use $\mathcal{D} = \{D_n \mid n \in N\}$ to denote an ensemble of probability distributions such that D_n is a distribution on I_n for each $n \in N$. Let P_f be a program that purports to compute a function f. We use $\mathrm{Err}(P_f, f, D_n)$ to denote the error probability that $P_f(x) \neq f(x)$ when x is randomly chosen in I_n according to D_n for $n \in N$. Let $0 < \beta < 1/2$ be a confidence parameter.

Definition 2.3 [BLR]: *A probabilistic polynomial time oracle program ST_f is said to be an $(\varepsilon_1, \varepsilon_2)$-*self-tester *for a function f with respect to $\mathcal{D} = \{D_n \mid n \in N\}$ if for some constants $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$ and any program P_f that purports to compute f, ST_f on input 1^n, $0 < \beta < 1/2$ satisfies the following conditions*:

(1) *If* $\mathrm{Err}(P_f, f, D_n) \leq \varepsilon_1$, *then* $\Pr\{ST_f^{P_f}(\langle 1^n, \beta\rangle) = \text{"pass"}\} \geq 1 - \beta$;

(2) *If* $\mathrm{Err}(P_f, f, D_n) \geq \varepsilon_2$, *then* $\Pr\{ST_f^{P_f}(\langle 1^n, \beta\rangle) = \text{"fail"}\} \geq 1 - \beta$,

where the probabilities are taken over all possible coin tosses of ST_f.

Note that in Definition 2.3, the output of ST_f is not specified when $\varepsilon_1 < \mathrm{Err}(P_f, f, D_n) < \varepsilon_2$. Thus the value $\delta = \varepsilon_2 - \varepsilon_1$ should be as close as possible to 0 so that ST_f is reliable.

We say that a function f is weakly/strongly *self-testable* (or f has a weak/strong self-tester) if for some/any ensemble of distributions $\mathcal{D}$, there exist some constants $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$ such that f has an $(\varepsilon_1, \varepsilon_2)$-self-tester ST_f with respect to $\mathcal{D}$ and we also say that a language L is weakly/strongly *self-testable* (or L has a weak/strong self-tester) if for some/any ensemble of distributions $\mathcal{D}$, there exists some constants $0 \leq \varepsilon_1 < \varepsilon_2 \leq 1$ such that its characteristic function λ_L has an $(\varepsilon_1, \varepsilon_2)$-self-tester ST_L with respect to $\mathcal{D}$. A function f (or a language L) is said to be strongly/weakly self-untestable if it is not weakly/strongly self-testable.

Definition 2.4 [BLR]: *A probabilistic polynomial time oracle program SC_f is said to be an ε-*self-corrector *for a function f with respect to $\mathcal{D} = \{D_n \mid n \in N\}$ if for some constant $0 < \varepsilon < 1$ and any program P_f that purports to compute f, SC_f on input 1^n, $x \in I_n$, $0 < \beta < 1/2$ satisfies the following condition*: *If* $\mathrm{Err}(P_f, f, D_n) \leq \varepsilon$, *then* $\Pr\{SC_f^{P_f}(\langle 1^n, x, \beta\rangle) = f(x)\} \geq 1 - \beta$.

We also say that a function f is weakly/strongly correctable (or f has a weak/strong self-tester), that a language L is weakly/storngly correctable (or L has a weak/storng self-tester), and that a function f (resp. a language L) is strongly/weakly self-uncorrectable in a way similar to the case of self-testers.

Definition 2.5 [BLR]: *A pair of probabilistic polynomial time oracle programs $\langle ST_f, SC_f\rangle$ is said to be a* self-tester/corrector pair *for a function f if for some constants $0 \leq \varepsilon_1 < \varepsilon_2 \leq \varepsilon < 1$ and an ensemble of distributions $\mathcal{D}$, ST_f is an $(\varepsilon_1, \varepsilon_2)$-self-tester for f with respect to $\mathcal{D}$ and SC_f is an ε-self-corrector for f with respect to $\mathcal{D}$.*

We say that a function f (resp. a language L) has a self-tester/corrector pair if there exists a self-tester/corrector pair for f (resp. the characteristic function λ_L of L).

For each $n \geq 0$, let $S_n = \{0,1\}^n$ and let R_n be a set of all possible sequences of coin tosses of a probabilistic polynomial (in n) time algorithm on input of length n.

Definition 2.6 (Oneway Permutation): *If a function $g : \{0,1\}^* \mapsto \{0,1\}^*$ satisfies that*

(1) *For any $x \in \{0,1\}^*$, $|g(x)| = |x|$ and g is 1-1 and onto;*

(2) *For any $x \in \{0,1\}^*$, $g(x)$ can be evaluated in polynomial (in $|x|$) time;*

(3) *For any probabilistic polynomial (in n) time algorithm A and each constant $c > 0$, there exists a constant $N_c \geq 0$ such that $\Pr\{g(A(y)) = y\} < n^{-c}$ for every $n > N_c$ and every $y \in \{0,1\}^n$, where the probabilities are taken over $\langle y, r\rangle \in S_n \times R_n$,*

then we say that the function $g : \{0,1\}^* \mapsto \{0,1\}^*$ is a oneway permutation.

In the rest of this paper, we sometimes use $g_n : \{0,1\}^n \mapsto \{0,1\}^n$ for each $n \in N$ to denote the restriction of a oneway permutation $g : \{0,1\}^* \mapsto \{0,1\}^*$.

The following inequality is useful in section 5.1 to approximate the error probability (with respect to $\mathcal{D}$) of a program P_f that purports to compute f.

Definition 2.7 [S]: *Let $X_1, X_2, \ldots, X_n$ be independent identically distributed 0-1 random variables and let $\Pr(X_i = 1) = p$ for each i $(1 \leq i \leq n)$. Then*

(1) $\Pr\left\{\frac{1}{n}\sum_{i=1}^{n} X_i - p \leq \delta\right\} \geq 1 - \exp\left\{-\frac{\delta^2 n}{2}\right\}$;

(2) $\Pr\left\{\frac{1}{n}\sum_{i=1}^{n} X_i - p \geq -\delta\right\} \geq 1 - \exp\left\{-\frac{\delta^2 n}{2}\right\}$,

for any integer $n \geq 1$ and any constant $\delta > 0$.

2.2 Known Results

Here we overview the known results on checkers, self-testers, and self-correctors. The following theorems are the relationships among checkers, self-testers, and self-correctors.

Theorem 2.8 [BLR], [R]: *If a function $f : \{0,1\}^* \mapsto \{0,1\}^*$ has a checker C_f, then for any constant $0 < \varepsilon_2 \leq 1$, the function f has a $(0, \varepsilon_2)$-self-tester ST_f with respect to any polynomial time samplable ensemble of distributions $\mathcal{D}$.*

Theorem 2.9 [BLR]: *If a function $f : \{0,1\}^* \mapsto \{0,1\}^*$ has a self-tester/corrector pair $\langle ST_f, SC_f\rangle$, then the function f has a checker C_f.*

3 Checkable But Self-Uncorrectable Languages

In this section, we give a solution to Question 1 in subsection 1.1, i.e., there exists a weakly self-uncorrectable language L that is checkable under a complexity-theoretic assumption.

Theorem 3.1: *If oneway permutations exist, then there exists a* weakly self-uncorrectable *language $L \notin \mathcal{BPP}$ that has a* checker.

Proof: Let $I = \{\{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \mid n \in N\}$ and let $I_n = \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$ for each $n \in N$. We use $\mathcal{U} = \{U_n \mid n \in N\}$ to denote an ensemble of uniform distributions, i.e., for each $n \in N$, U_n is a uniform distribution on I_n, and use $a \cdot b$ to denote the inner product of $a, b \in \{0,1\}^n$ modulo 2. Let $g : \{0,1\}^* \mapsto \{0,1\}^*$ be a oneway permutation. Define a language $L_g \subseteq I$ to be $L_g = \{\langle y, p, 0^{|y|}\rangle \in I \mid g^{-1}(y) \cdot p = 1\}$. It is obvious that $L_g \in \mathcal{NP}$. Note that the definition of L_g is inspired by the hard-core predicate due to Goldreich and Levin [GL].

We first show that L_g has a checker C_g. But there isn't enough space to show that. So we obmit the proof.

Next, we show that L_g is weakly self-uncorrectable. Indeed, we show that for some ensemble of distributions $\mathcal{D} = \{D_n \mid n \in N\}$ and for any constant $0 < \varepsilon < 1$, the characteristic function λ_g of L_g does not have an ε-self-corrector with respect to $\mathcal{D} = \{D_n \mid n \in N\}$.

Assume that L_g is **strongly self-correctable**, i.e., for any ensemble of distributions $\mathcal{D}$ and some constant $0 < \varepsilon < 1$, L_g has an ε-self-corrector with respect to $\mathcal{D}$. Then this means that for some constant $0 < \varepsilon < 1$, L_g has an ε-self-corrector SC_g^ε with respect to $\mathcal{U} = \{U_n \mid n \in N\}$, where U_n is a uniform distribution on I_n for each $n \in N$. Define a program $\tilde{P}_g$ that purports to cumpute λ_g in a way that on input $\langle y, p, z \rangle \in I_n$,

$$\tilde{P}_g(\langle y,p,z\rangle) = \begin{cases} 0 & z \neq 0^n; \\ 1 & z = 0^n. \end{cases}$$

Here we note that $\tilde{P}_g$ errs only for $\langle y, p, 0^n \rangle \notin L_g \cap I_n$. From the assumption that g_n is a (oneway) permutation, it follows that for each $n \in N$, $\mathrm{Err}(\tilde{P}_g, \lambda_g, U_n)$ is given by

$$\mathrm{Err}(\tilde{P}_g, \lambda_g, U_n) = \frac{\|\{\langle y,p,0^n\rangle \in I_n \mid g^{-1}(y)\cdot p = 0\}\|}{\|I_n\|} = \frac{2^{2n-1}+2^{n-1}}{2^{3n}} = \frac{2^n+1}{2^{2n+1}} < \frac{1}{2^n},$$

where $\|A\|$ denotes the cardinality of a finite set A.

Then for every $n \geq \lceil \log \varepsilon^{-1} \rceil$, $\mathrm{Err}(\tilde{P}_g, \lambda_g, U_n) \leq \varepsilon$. This implies that for every $n \geq \lceil \log \varepsilon^{-1} \rceil$, SC_g^ε making calls to the program $\tilde{P}_g$ outputs 1 with probability at least $1 - \beta$ if $\langle y, p, z \rangle \in L_g$ and outputs 0 with probability at least $1 - \beta$ if $\langle y, p, z \rangle \notin L_g$. Thus we can make probabilistic polynomial (in n) time program $\widehat{P}_g$ such that $\widehat{P}_g(\langle y,p,z\rangle) = \lambda_g(\langle y,p,z\rangle)$ with probability at least $1-\beta$ from C_g and $\tilde{P}_g$. Since $0 < \beta < 1/2$ is a constant (see Definition 2.4), $L_g \in \mathcal{BPP}$ [BDG]. By a standard technique (see [BDG]), we assume without loss of generality that there exists a probabilistic polynomial (in n) time algorithm G such that $G(\langle y,p,z\rangle) = \lambda_g(\langle y,p,z\rangle)$ with probability at least $1 - 2^{-n}$. Then let us consider the following probabilistic algorithm Inv_g:

`Inverting Algorithm` Inv_g:

Input: $y \in \{0,1\}^n$.

I-0: $x := \varepsilon$ (null string); $j := 1$.

I-1: Choose randomly $p \in \{0,1\}^n$.

I-2: Run G on input $\langle y, p, 0^n \rangle$ to get $b = G(\langle y, p, 0^n \rangle)$.

I-3: Run G on input $\langle y, p \oplus e_j^n, 0^n \rangle$ to get $b_j = G(\langle y, p \oplus e_j^n, 0^n \rangle)$.

I-4: If $b = b_j$, then $x := x\|0$ and $j := j+1$; otherwise $x := x\|1$ and $j := j+1$.

I-5: If $j \leq n$, then go to step I-3; otherwise continue.

I-6: If $y = g(x)$, then halt and output $x \in \{0,1\}^n$; otherwise halt and output "$\perp$."

It is obvious that if G correctly returns b and b_j for each j $(1 \leq j \leq n)$, then Inv_g successfully finds in polynomial time $x \in \{0,1\}^n$ such that $y = g(x)$ (see [GL]). Thus the probability P_{succ} that on input $y \in \{0,1\}^n$, Inv_g outputs $x \in \{0,1\}^n$ such that $y = g(x)$ is bounded by

$$P_{succ} \geq \left(1 - 2^{-n}\right)^{n+1} \geq 1 - (n+1)\cdot 2^{-n}.$$

This contradicts the assumption that g is a oneway permutation (see Definition 2.6). Then it follows that $L_g \notin \mathcal{BPP}$ and this implies that L_g must be weakly self-uncorrectable.

Thus if oneway permutations exist, then there exists a weakly self-uncorrectable language $L \notin \mathcal{BPP}$ that has a different checker. ∎

4 Self-Debuggers

In this section, we present a new notion of "self-debuggers." Informally, a function f is said to have a self-debugger SD_f with respect to $\mathcal{D}$ if SD_f transforms a faulty program P_f for f with respect to $\mathcal{D}$ to a less faulty (deterministic) program Q_f for f with respect to $\mathcal{D}$.

Definition 4.1: *A probabilistic polynomial time oracle program SD_f is said to be a (δ,γ)-* **self-debugger** *for a function f with respect to $\mathcal{D} = \{D_n \mid n \in N\}$ if for some constants $0 \le \gamma \le \delta \le 1$, SD_f on input 1^n, $0 < \beta < 1/2$ transforms any program P_f that purports to compute f to a (deterministic) program Q_f^n for f and it satisfies the following conditions:*

(1) *If* $\mathrm{Err}(P_f, f, D_n) \le \delta$, *then* $\Pr\{\mathrm{Err}(Q_f^n, f, D_n) \le \gamma\} \ge 1 - \beta$;

(2) *If* $\mathrm{Err}(P_f, f, D_n) > \delta$, *then* $\Pr\{SD_f^{P_f}(\langle 1^n, \beta\rangle) = \text{"fail"} \vee \mathrm{Err}(Q_f^n, f, D_n) \le \gamma\} \ge 1 - \beta$,

where the probabilities are taken over all possible coin tosses of SD_f

The following theorem shows that if a function f has a self-tester/corrector pair, then there exists a self-debugger for f.

Theorem 4.2: *For some constants $0 < \varepsilon_1' < \varepsilon_2' \le \varepsilon' < 1$ and a polynomial time samplable ensemble of distributions $\mathcal{D}$, let $\langle \overline{ST}_f, \overline{SC}_f\rangle$ is a self-tester/corrector pair for a function f, i.e., $\overline{ST}_f$ is an $(\varepsilon_1', \varepsilon_2')$-self-tester for f with respect to $\mathcal{D}$ and $\overline{SC}_f$ is an ε'-self-corrector for f with respect to $\mathcal{D}$. Then there exists an $(\varepsilon', \varepsilon_2')$-* **self-debugger** *$SD_f$ for f with respect to $\mathcal{D}$.*

5 Applications of Self-Debuggers

In this section, we present two applications of self-debuggers by using Theorem 4.2. The first application is to design a reliable and flexible self-tester from any self-tester/corrector pair and the second application is to design strong self-debugger from any self-tester/corrector pair.

5.1 Making Self-Testers Reliable and Flexible

Assume that a self-tester/corrector pair $\langle \overline{ST}_f, \overline{SC}_f\rangle$ for a function $f : \{0,1\}^* \mapsto \{0,1\}^*$ is given. Then it follows from Theorem 2.9 that f has a checker C_f and it follows from Theorem 2.8 that for any constant $0 < \varepsilon_2 \le 1$, f has a $(0, \varepsilon_2)$-self-tester $ST_f^{\varepsilon_2}$ with respect to any polynomial time samplable ensemble of distributions $\mathcal{D}$. The resulting self-tester $ST_f^{\varepsilon_2}$ can be highly reliable when ε_2 is taken to be very small. On the other hand, $ST_f^{\varepsilon_2}$ is somewhat of limited use, because it only passes completely correct programs P_f for f. We often wish to design a flexible self-tester ST_f for f in such a way that ST_f passes almost correct programs $\tilde{P}_f$ for f. In this section, we give (as a solution to Question 2) a way to design a reliable and flexible self-tester for f from any self-tester/corrector pair for f.

Theorem 5.1: *For some constants $0 < \varepsilon_1' < \varepsilon_2' \le \varepsilon' < 1$ and a polynomial time samplable ensemble of distributions $\mathcal{D}$, let $\langle \overline{ST}_f, \overline{ST}_f\rangle$ is a self-tester/corrector pair for a function f, i.e., $\overline{ST}_f$ is an $(\varepsilon_1', \varepsilon_2')$-self-tester for f with respect to $\mathcal{D}$ and $\overline{SC}_f$ is an ε'-self-corrector for f with respect to $\mathcal{D}$. Then for any constants $\varepsilon_1 < \varepsilon_2$ such that $\varepsilon_1 \le \varepsilon'$, there exists an $(\varepsilon_1, \varepsilon_2)$-* **self-tester** *$ST_f$ for f with respect to $\mathcal{D}$.*

5.2 Amplification of Self-Debuggers

From Theorems 4.2 and 5.1, we can show the following theorem:

Theorem 5.2:

*For some constants $0 < \varepsilon_1' < \varepsilon_2' \leq \varepsilon' < 1$ and a polynomial time samplable ensemble of distributions $\mathcal{D}$, let Let $0 < \varepsilon_1' < \varepsilon_2' \leq n\varepsilon' < 1$ be constants and let $\mathcal{D}$ be a polynomial time samplable ensemble of distributions. $\langle \overline{ST}_f, \overline{ST}_f \rangle$ is a self-tester/corrector pair for a function f, i.e., $\overline{ST}_f$ is an $(\varepsilon_1', \varepsilon_2')$-self-tester for f with respect to $\mathcal{D}$ and $\overline{SC}_f$ is an ε'-self-corrector for f with respect to $\mathcal{D}$. Then for any constant $\delta > 0$ such that $\delta \leq \varepsilon'$, there exists an (ε', δ)-***self-debugger*** *SD_f for f with respect to $\mathcal{D}$.*

Proof: This can be led from Theorem 5.1 and Theorem 4.2 easily.

Acknowledgments

The authors wish to thank Ronitt Rubinfeld for her helpful discussion on the relation between checkers and self-testers [R].

References

[BDG] Balcázar, J.L., Díaz, J., and Gabarró, J., "Structural Complexity I," EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin (1988).

[BF] Beigel, R. and Feigenbaum, J., "On Being Coherent Without Being Very Hard," *Computational Complexity*, Vol.2, No.1, pp.1-17 (1992).

[BG] Bellare, M. and Goldwasser, S., "The Complexity of Decision versus Search," to appear in *SIAM Journal on Computing.*

[BK] Blum, M. and Kannan, S., "Designing Programs That Check Their Work," Proceedings of STOC'89, pp.86-97 (1989).

[BLR] Blum, M., Luby, M., and Rubinfeld, R., "Self-Testing/Correcting with Applications to Numerical Problems," *Journal of Computer and Systems Sciences*, Vol.47, No.3, pp.549-595 (1993).

[BM] Babai, L. and Moran, S., "Arthur-Merlin Games: A Randomized Proof Systems and a Hierarchy of Complexity Classes," *Journal of Computer and System Sciences*, Vol.36, pp.254-276 (1988).

[GL] Goldreich, O. and Levin, L.A., "A Hard-Core Predicate for All One-Way Functions," Proceedings of STOC'89, pp.25-32 (1989).

[GMR] Goldwasser, S., Micali, S., and Rackoff, C., "The Knowledge Complexity of Interactive Proof Systems," *SIAM Journal on Computing*, Vol.18, No.1, pp.186-208 (1989).

[R] Rubinfeld, R., private communication (1993).

[S] Spencer, J., "Ten lectures on the Probabilistic Method," CBMS-NSF Regional Conference Series in Applied Mathematics 52, SIAM (1987).

[Y] Yao, A., "Coherent Functions and Program Checkers," Proceedings of STOC'90, pp.84-94 (1990).