THEORY OF COMPUTING, Volume 2 (2006), pp. 173–183
http://theoryofcomputing.org

Tolerant Versus Intolerant Testing for Boolean Properties

Eldar Fischer* Lance Fortnow

Received: April 3, 2006; published: September 25, 2006.

Abstract: A property tester with high probability accepts inputs satisfying a given property and rejects inputs that are far from satisfying it. A tolerant property tester, as defined by Parnas, Ron and Rubinfeld, must also accept inputs that are close enough to satisfying the property. We construct two properties of binary functions for which there exists a test making a constant number of queries, but yet there exists no such tolerant test. The first construction uses Hadamard codes and long codes. Then, using Probabilistically Checkable Proofs of Proximity as constructed by Ben-Sasson et al., we exhibit a property which has constant query intolerant testers but for which any tolerant tester requires $n^{\Omega(1)}$ queries.

ACM Classification: G.3, F.2.2

AMS Classification: 68Q99, 68W20

Key words and phrases: Property Testing, Tolerant Testing, PCP of Proximity

1 Introduction

Combinatorial property testing deals with the following task: For a fixed $\varepsilon > 0$ and a fixed property R, distinguish using as few queries as possible (with high confidence) between the case that an input of length m satisfies R, and the case that the input is ε-far from satisfying R, i. e., the input differs in at least an ε-fraction of the bits from every string satisfying R. In our context the inputs are Boolean, and the distance from R is measured by the minimum number of bits that have to be modified in the input to make it satisfy R, divided by the input length m. For the purpose here we are mainly interested in tests

*Research supported in part by an Israel Science Foundation grant number 55/03.

 DOI: 10.4086/toc.2006.v002a009

that have a number of queries that depends only on the approximation parameter ε and is independent of the input length. Properties that admit such algorithms are called *testable*.

Blum, Luby, and Rubinfeld [7] were the first to investigate a question formulated in terms of property testing, and Rubinfeld and Sudan [22] formally defined the general notion of property testing. Goldreich, Goldwasser, and Ron [16] investigated property testing in the combinatorial context, where they first formalized the testing of combinatorial objects such as graphs. In recent years the field of property testing has enjoyed rapid growth, as witnessed in the surveys of Ron [21] and Fischer [11].

Since even a correct input may have a small amount of noise, Parnas, Ron, and Rubinfeld [19] have recently started investigating property testing algorithms which are guaranteed to accept (with high confidence) not only inputs that satisfy the property, but also inputs that are sufficiently close to satisfying it. The following formal definition highlights this distinction.

Definition 1.1. Given a property R, an ε*-test* for R is a randomized algorithm that is guaranteed to accept with probability at least $\frac{2}{3}$ any input that satisfies R, and is guaranteed to reject with probability at least $\frac{2}{3}$ any input that is ε-far from satisfying R. We say that the property is *testable* if for every $\varepsilon > 0$ there exists an ε-test whose number of queries is independent of the input size m.

A 1*-sided* ε*-test* for R is an ε-test as above that in addition is guaranteed to accept any input that satisfies R with probability 1.

A *tolerant* (ε,δ)*-test* for R is an ε-test for R that in addition is guaranteed to accept with probability at least $\frac{2}{3}$ any input that is δ-close to satisfying R, where an input is said to be δ*-close to satisfying* R if it is not δ-far from satisfying R. We say that a property is *tolerantly testable* if for every $\varepsilon > 0$ there exists a constant $\delta > 0$ for which there exists a (ε,δ)-test whose number of queries is independent of m.

Many properties that are testable as per the definition above are also tolerantly testable. Alon et al. [1] implicitly give tolerant tests for the testable graph properties, and such tests also follow from the canonical testing result of Goldreich and Trevisan [17]. Fischer and Newman [14] prove an even stronger result that every testable graph property is also (ε,δ)-testable for *any* $\delta < \varepsilon$, showing that for the dense graph model testability in fact implies that the distance of an input graph from a property can be estimated using a number of queries that depends only on the additive approximation term.

For non-Boolean properties there are easy examples of properties where the number of queries required for an ε-test may be much smaller than the number required for an (ε,δ)-test. Consider the following example that uses bounds on testing invertability and inverseness of functions, implicit in the works of Ergün et al. [9] and Ergün, Kumar, and Rubinfeld [10] about testing for element distinctness and multiset equality. Consider the property of a sequence of n^2 numbers consisting of (the representation of) $n-1$ copies of a function $f : \{1,\dots,n\} \to \{1,\dots,n\}$ and one copy of its inverse function g. An easy test follows from uniformly sampling values i and checking that indeed $f(g(i)) = g(f(i)) = i$ (as well as sampling from the supposed $n-1$ copies of f and checking that they agree with each other on i). On the other hand, a tolerant test would have to ignore the representation of g altogether because the encoding of g makes up only a tiny part of the total input, and testing whether a function f has some inverse is hard.

If we try to directly convert such examples to properties of Boolean functions, for example by taking the Boolean representation of the values of f and g, then with some tweaking we can see a difference in the number of required queries between a tolerant and an intolerant test, but it will typically be between two different constants. This still leaves open the question of whether every property, for which there exists a (constant query complexity) ε-test for every $\varepsilon > 0$, admits also constant query complexity tolerant tests. In this paper we prove that this is not the case, and construct properties that admit intolerant tests with a constant number of queries but admit no such tolerant tests.

Theorem 1.2. *There exists a property R, such that for every ε there exists an ε-test for R making a number of queries that depends only on ε (and not on the input size), while for every constant $0 < \delta < \frac{1}{4}$ and q there exists no tolerant $(\frac{1}{4}, \delta)$-test making only q queries (for large enough inputs).*

The proof of the above combines results from several topics of property testing, including one of the very first results in this field, linearity testing [7]. Alternatively, using the recently constructed Probabilistically Checkable Proofs of Proximity by Ben-Sasson et al. [5] we can prove a strengthening of Theorem 1.2.

Theorem 1.3. *There exists a property R, such that for every ε there exists an ε-test for R making a number of queries that depends only on ε (and not on the input size), while there exists a constant $c > 0$ such that for every constant $0 < \delta < \frac{1}{4}$ there exists no tolerant $(\frac{1}{4}, \delta)$-test making only n^c queries (for large enough inputs).*

The proof of Theorem 1.3 relies on the heavy machinery of Probabilistically Checkable Proofs. We present its proof following a separate direct proof of Theorem 1.2 (which, if analyzed carefully, would have given an $\Omega(\log\log n)$ lower bound on the query complexity).

The rest of the paper is organized as follows. In Section 2 we present the basic building blocks for the proof of Theorem 1.2, for which we need results that were proven all throughout the history of the field, and in Section 3 we string them together proving Theorem 1.2. Section 4 contains the proof of Theorem 1.3, which gives better lower bounds but requires less direct methods.

These results originally appeared in the Proceedings of the 20th IEEE Conference on Computational Complexity [12].

2 Preliminaries

We base our first property on Hadamard codes and long codes. In the following we somewhat abuse notation, and when clear from context refer by the word "code" both to a legal codeword, and to the set of all allowed codewords. In particular, the term "a property of a code" refers to a subset of the set of codewords. In the following we will use the fact that some properties of codes are testable (i. e. there exists an ε-test for the property in the usual sense if we know in advance that the input is a legal codeword), while other properties of codes are not testable.

A *Hadamard code* is a string x of length 2^n, for which there exists a string y of length n such that for every i the ith bit of x is equal to $y \cdot i$ (where we use the binary representation of i, and the "dot product" is defined over $\mathbb{Z}_2$ as $a \cdot b = \bigoplus_{j=1}^{n} a_j b_j$). Equivalently, a string x is a Hadamard code if and only if $f(i) = x_i$ is a linear function over $\mathbb{Z}_2$. We shall use the two definitions interchangeably.

Let $f_1, \ldots, f_{2^{2^n}}$ be an enumeration of all of the functions on inputs of length n, according to the lexicographic order on the sequence of their values on the domain $\{0,1\}^n$. A *long code* is a string x of length 2^{2^n} such that $x_i = f_i(y)$ for every j for some fixed y of length n. Here too there is a useful equivalence.

The string x is a long code if and only if $g(i) = x_i$ is a dictator function, i. e., when there exists a j for the above $g : \{0,1\}^{2^n} \to \{0,1\}$ such that for all $z \in \{0,1\}^{2^n}$, $g(z) = z_j$ (note that in particular a long code is also linear). We get the correspondence by setting $g(i) = f_i(j)$. The extreme redundancy of long codes has proven itself to be very useful in complexity theory, for example in the optimal inapproximability results of Håstad [18].

The possibility for testing that a function is a Hadamard code in fact stems from one of the very first results in the field of property testing.

Lemma 2.1 ([7]). *For every ε, the property that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is linear (over the field $\mathbb{Z}_2$) is testable with a 1-sided test using a number of queries that depends only on ε.*

Since the property that a function $h : \{0,1\}^n \to \{0,1\}$ is a Hadamard code of some $y = b_1,\dots,b_n$ is identical to the property of h being linear over $\mathbb{Z}_2$, we can use the above for testing this. Testing for long codes follows from somewhat more recent results.

Lemma 2.2 ([4, 20]). *For every ε, the property that a Boolean function $f : \{0,1\}^m \to \{0,1\}$ is a dictator function is testable with a 1-sided test using a number of queries that depends only on ε.*

Properties of long codes of binary strings can be easily tested for, since a proper long code of a string contains its corresponding value for every possible function. Thus, given a code L, we can test it by looking at its value for the function that describes the tested property itself. Further details are provided in the proof of Lemma 3.1 below.

On the other hand, there exist properties of Hadamard codes that are hard to test. Such properties have been used to prove the existence of properties that can be easily tested only using a quantum algorithm, by Buhrman, Fortnow, Newman, and Röhrig [8], and another property of Hadamard codes with additional features was implicitly used also by Fischer et al. [13].

Lemma 2.3 ([8]). *There exist properties of Hadamard codes that cannot be $\frac{1}{3}$-tested (even by a 2-sided test) with a constant number of queries.*

The work of Fischer et al. [13] implies that one cannot distinguish with a constant number of queries between a linear Boolean function depending on exactly $\lfloor \frac{1}{2}n \rfloor$ variables and one that depends on exactly $\lfloor \frac{1}{2}n \rfloor + 2$ variables, and so the property of being a Hadamard code of a string with exactly $\lfloor \frac{1}{2}n \rfloor$ nonzero bits is not testable.

We use such a property of a Hadamard code because it will always yield an easy "long-code assisted test," despite the Hadamard code being hard to test in an "unassisted" manner. In essence, our input is supposed to contain a Hadamard code and a long code that extends the Hadamard code (i. e. both codes encode the same "original value"). Using the discussion above together with the self-correction features of Hadamard codes and long codes, we will show how to create a test by checking the Hadamard code against the long code, and then testing the long code for our property. However, we cannot test the Hadamard code alone if we are not allowed to look at the long code.

The notion of "assisted tests" reminds one of the essence of the work of Ergün, Kumar, and Rubinfeld [10] and Batu, Rubinfeld, and White [3], only here the "witness" can have exponential size because we can do weighting by replication. For the construction with the better lower bounds, we will use a strong result of Ben-Sasson et al. [5] about assisted tests.

With all the above components in hand, we are now ready to construct a property that has an easy test but not a tolerant one.

3 Proof of Theorem 1.2

In the following, for a parameter n, we consider inputs whose size is $(2^n+1)2^{2^n}$. We consider the input as composed of one function L from the set of functions $\{f \mid f : \{0,1\}^n \to \{0,1\}\}$ to $\{0,1\}$ (the function L takes 2^{2^n} bits to write down), and $l = 2^{2^n}$ functions $h_1,\dots,h_l$ from $\{0,1\}^n$ to $\{0,1\}$ (each such function takes 2^n bits to write down), where all functions are represented by their truth tables.

We pick a property U of Hadamard codes that satisfies Lemma 2.3, and define *Property R* as the property of the input satisfying the following: *All the functions $h_1,\dots,h_l$ are identical and are equal to a Hadamard code of some $x \in \{0,1\}^n$ that satisfies property U, and the function L is the long code of this same x.*

Lemma 3.1. *Property R admits a* 1*-sided ε-test with a constant number of queries for every ε.*

Proof. We assume that $\varepsilon < \frac{1}{8}$, and do the following.

- Repeating independently $100\varepsilon^{-1}$ times, we select a uniformly random $x \in \{0,1\}^n$, a uniformly random $1 \le i \le l$, and check that the bit corresponding to $h_1(x)$ is indeed equal to that of $h_i(x)$. If any of these checks fails, we reject the input.

- Using Lemma 2.1 we perform a $\frac{1}{2}\varepsilon$-test of $h_1(x)$ for the property of being a linear function (i. e. being a Hadamard code of some $b_1,\dots,b_n$). We amplify the success probability of the test to $\frac{19}{20}$, so that the probability of a false positive answer will be no greater than $\frac{1}{20}$.

- Using Lemma 2.2 we perform an ε-test of $L(f)$ for the property of being a long code of some $x \in \{0,1\}^n$. Again we amplify the success probability of the test to $\frac{19}{20}$.

- Denote for any $y \in \{0,1\}^n$ by $\chi_y : \{0,1\}^n \to \{0,1\}$ the corresponding Hadamard code (i. e. for $y = (a_1,\dots,a_n)$, we set $\chi_y(b_1,\dots,b_n) = \bigoplus_{i=1}^n a_i b_i$). We perform 100 iterations of the following: We select a uniformly random $y \in \{0,1\}^n$, a uniformly random $f : \{0,1\}^n \to \{0,1\}$, and check that $h_1(y) = L(f) \oplus L(f \oplus \chi_y)$, rejecting the input if any of the checks fail.

- Now let $u(x) : \{0,1\}^n \to \{0,1\}$ denote the indicator function of Property U, i. e. $u(x) = 1$ if and only if the Hadamard code of x satisfies Property U. We now perform 100 iterations of choosing a uniformly random $f : \{0,1\}^n \to \{0,1\}$, and checking that $L(f) \oplus L(f \oplus u) = 1$, rejecting if any of these checks fail.

On one hand, it is clear that an input that satisfies Property R will be accepted with probability 1. On the other hand, if an input is accepted with probability at least $\frac{2}{3}$, then all of the following hold.

- The portion of the input that corresponds to $h_2(x),\dots,h_l(x)$ is $\frac{1}{2}\varepsilon$-close to being $l-1$ copies of the function $h_1(x)$.

- $h_1(x)$ is $\frac{1}{2}\varepsilon$-close to being the Hadamard code of some $(b_1,\ldots,b_n) \in \{0,1\}^n$. With the previous item this means that the restriction of the input to $h_1(x),\ldots,h_l(x)$ is ε-close to being l copies of the Hadamard code of $b_1,\ldots,b_n$.

- $L(f)$ is ε-close to being a long code of some $(c_1,\ldots,c_n) \in \{0,1\}^n$.

- $(b_1,\ldots,b_n) = (c_1,\ldots,c_n)$. Otherwise every iteration of the check in the fourth item above would fail with probability at least $\frac{1}{8}$. This is since doing such a check between an actual Hadamard code and an actual long code of differing strings would fail with probability $\frac{1}{2}$; the additional loss of $\frac{3}{8}$ in the probability is because $h_1(x)$ is only guaranteed to be $\frac{1}{16}$ close to being the Hadamard code of $b_1,\ldots,b_n$, and $L(f)$ is only guaranteed to be $\frac{1}{8}$-close to the long code of $c_1,\ldots,c_n$.

- $b_1,\ldots,b_n$ satisfy Property U (and with the above items this means that the input as a whole is in fact ε-close to satisfying Property R). The reason is that otherwise every iteration of the check in the fifth item of the test would fail with probability at least $1-2\varepsilon > \frac{3}{4}$.

The above complete the proof of the test. □

Lemma 3.2. *There exist no constants δ and q, for which property R can be $(\frac{1}{4},\delta)$-tested for every n using only q queries.*

Proof. We may assume that $\delta < \frac{1}{12}$. We show that if there exists a $(\frac{1}{4},\delta)$-test for R, then for every large enough n there exists a $\frac{1}{3}$-test for U (not necessarily a tolerant one) making only q queries, which is known not to exist by Lemma 2.3.

Given an input $h:\{0,1\}^n \to \{0,1\}$ which we would like to test for Property U, we construct an input for Property R as follows: $h_1,\ldots,h_l$ will all be identical to h, and L will be arbitrarily set to the all-zero function. Note that any single query to the new input can be answered by making a single query (or no query) to the original input.

The next thing to note is that for n large enough, if h satisfies U then the new input is δ-close to satisfying R, because for n large enough the number of bits in the function L is less than a δ-fraction of the total number of bits in the input, while $h_1,\ldots,h_l$ clearly satisfy all requirements not concerning L in the definition of R. On the other hand, if the new input is $\frac{1}{4}$-close to satisfying Property R, then h is necessarily $\frac{1}{3}$-close to satisfying Property U, because of what the definition of Property R states for $h_1,\ldots,h_l$. We thus obtain our $\frac{1}{3}$-test for U. □

The above two lemmas complete the proof of Theorem 1.2.

4 PCPs of Proximity and Theorem 1.3

This section gives a proof of Theorem 1.3 that strengthens Theorem 1.2. We first define the construction and cite the main lemma that we will use.

Property testing has some common origins with Probabilistically Checkable Proofs, and Ergün et al. [10] and Batu et al. [3] investigated this connection further, with regards to using a PCP witness for an input.

Definition 4.1. Given a promise problem and a Boolean input $v_1, \dots, v_n$, a (1-sided) *PCP witness* for the problem is a set of functions $f_1, \dots, f_l$, where l is polynomial in n, satisfying the following.

- Each of the functions has a number of variables bounded by a constant independent of n, that may include variables from $v_1, \dots, v_n$ as well as from an additional set of (polynomially many) Boolean variables $w_1, \dots, w_m$.
- If $v_1, \dots, v_n$ should be accepted according to the promise problem, then there exists an assignment to $w_1, \dots, w_m$ that together with $v_1, \dots, v_n$ satisfies all the functions $f_1, \dots, f_l$.
- If $v_1, \dots, v_n$ should be rejected according to the promise problem, then there exists no assignment to $w_1, \dots, w_m$ for which more than $\frac{1}{2}l$ of the functions are satisfied.

A *PCP of Proximity* is a PCP witness for the promise problem of accepting all inputs that satisfy a given property P, and rejecting all inputs that are ε-far from P for a given distance parameter ε.

A recent strong result, concerning the existence of PCPs of Proximity for all properties decidable in polynomial time, is given by Ben-Sasson et al. [5].

Lemma 4.2 (Special case of [5]). *If P is a property of $v_1, \dots, v_n$ that is decidable by a circuit of size k, and $t < \log\log k / \log\log\log k$, then there exists a PCP of Proximity for P with distance parameter $1/t$. Moreover, the number of additional variables and the number of functions are both bounded by k^2, and each function depends on $O(t)$ variables.*

On the other hand, there is a plethora of lower bound results for properties which belong to low complexity classes (e. g. [2, 6, 15]) and most of them would work fine for us. We will choose the property $U = \{uu^R vv^R \mid u, v \in \{0,1\}^*\}$, where w^R denotes the reversal of the word w.

Lemma 4.3 (Alon et al. [2]). *Property U can be computed in polynomial time, while any $\frac{1}{3}$-test for U requires at least $\Omega(\sqrt{n})$ queries (where n is the input size).*

We let $p(n)$ be a polynomial bound on the circuit size for deciding Property U on inputs of size n.

To construct the property to fulfill Theorem 1.3, we first assume without loss of generality that n divides $p(n)$ and set $t_n = \lfloor \log\log\log p(n) \rfloor$, so in particular $t_n < \log\log p(n)/\log\log\log p(n)$ for a sufficiently large n. We consider inputs of size $n(p(n))^2$. We label the first $(n - t_n)(p(n))^2$ bits by $(v_{i,j})_{1 \le i \le n, 1 \le j \le (n-t_n)(p(n))^2/n}$, and the rest of the bits by $(w_{i,j})_{1 \le i \le (p(n))^2, 1 \le j \le t_n}$. We define *Property R* as the set of inputs satisfying all of the following.

- For every i, $1 \le i \le n$, and j, $1 < j \le (n - t_n)(p(n))^2/n$, $v_{i,1} = v_{i,j}$ (so we have $(n - t_n)(p(n))^2/n$ copies of the same string).
- $v_{1,1}, \dots, v_{n,1}$ satisfy Property U.
- For every j, $1 \le j \le t_n$, $w_{1,j}, \dots, w_{(p(n))^2,j}$ is an assignment satisfying the PCP of Proximity for Property U (from Lemma 4.2) with distance parameter $1/j$, with regards to $v_{1,1}, \dots, v_{n,1}$.

We now prove that this is the required property.

Lemma 4.4. *Property R is (non-tolerantly) testable.*

Proof. For every ε we show how for n large enough we can ε-test for R using a constant number of queries (and for smaller n we can just read the entire input). We assume that $\varepsilon < \frac{1}{8}$ and that n is large enough to satisfy $t_n > 3/\varepsilon$, and do the following.

- Repeating independently $100\varepsilon^{-1}$ times, we select a uniformly random i, $1 \le i \le n$, a uniformly random j, $1 < j \le (n - t_n)(p(n))^2/n$, and check that $v_{i,1} = v_{i,j}$. If any of these checks fails, we reject the input.

- For $j = \lceil 3/\varepsilon \rceil$, for 100 iterations we select a uniformly random i, $1 \le i \le l$ (where l is the number of functions in the corresponding PCP of Proximity from Lemma 4.2), and each time test that the function f_i is satisfied by $v_{1,1}, \ldots, v_{n,1}$ and $w_{1,j}, \ldots, w_{(p(n))^2,j}$.

This test makes a constant number of queries, as the PCP of Proximity was invoked with a distance parameter that depends only on ε. It is also clear that if the input satisfies Property R, then it is accepted by this tester with probability 1.

On the other hand, if the input is satisfied with probability at least $\frac{1}{3}$, then $v_{1,1}, \ldots, v_{n,1}$ is $\frac{1}{3}\varepsilon$-close to some $v'_{1,1}, \ldots, v'_{n,1}$ satisfying Property U, and the rest of the $v_{i,j}$ are $\frac{1}{3}\varepsilon$-close to satisfying the equalities with $v_{1,j}$ and thus are $\frac{2}{3}\varepsilon$-close to being copies of the $v'_{1,1}, \ldots, v'_{n,1}$. But as the $w_{i,j}$ form less than a $\frac{1}{3}\varepsilon$ fraction of the total input size, this means that the input is ε-close to satisfying Property R. □

Lemma 4.5. *There exists some $c > 0$, so that there exists no δ for which Property R can be $(\frac{1}{4}, \delta)$-tested with n^c queries.*

Proof. We assume that $\delta < \frac{1}{12}$. Let $c_1 > 0$ be such that Property U cannot be $\frac{1}{3}$-tested with n^{c_1} queries, and let $c_2 > 0$ be such that $n(p(n))^2 < n^{1/c_2}$ for $n > 1$. We set $c = c_1 c_2$, and prove that a $(\frac{1}{4}, \delta)$-test with n^c queries for Property R implies (for all n large enough) a $\frac{1}{3}$-test with n^{c_1} queries for Property U, leading to a contradiction.

Given an input $v_1, \ldots, v_n$ which we would like to $\frac{1}{3}$-test, we construct an input of size $n(p(n))^2$ to test for Property R as follows. We set $v_{i,j} = v_i$ for all $1 \le i \le n$ and $1 \le j \le (n - t_n)(p(n))^2/n$, and arbitrarily set $w_{i,j} = 0$. As in Section 3, it is clear that a query to the new input can be simulated by performing at most one query to the original input. Also, for n large enough, if $v_1, \ldots, v_n$ satisfy Property U then the new input is δ-close to satisfying Property R (because the $w_{i,j}$ form less than a δ fraction of the input bits). On the other hand if the new input is $\frac{1}{4}$-close to satisfying Property R then the original input was $\frac{1}{3}$-close to satisfying Property U.

The above implies that a $(\frac{1}{4}, \delta)$-test for Property R that makes at most $(n(p(n))^2)^c < n^{c_1}$ queries would yield a $\frac{1}{3}$-test for Property U that makes at most n^{c_1} queries, a contradiction. □

The above two lemmas complete the proof of Theorem 1.3.

A concluding comment

Theorem 1.3 gives an example of a testable property for which there is an n^c lower bound for tolerant (ε,δ)-testing, for some fixed ε and any constant δ. It would be interesting to know whether there exists any (non-tolerantly) testable Boolean property for which any tolerant test requires a *linear* number of queries. Either a positive or a negative answer would likely have interesting effects, because of the connection explored here between tolerant testing and PCPs of Proximity.

Acknowledgments

We thank Prahladh Harsha for his help with Probabilistically Checkable Proofs of Proximity, particularly with the statement of Lemma 4.2.

References

[1] * N. ALON, E. FISCHER, M. KRIVELEVICH, AND M. SZEGEDY: Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000. [Combinatorica:mwapje2fdyk7ma2e]. 1

[2] * N. ALON, M. KRIVELEVICH, I. NEWMAN, AND M. SZEGEDY: Regular languages are testable with a constant number of queries. *SIAM Journal on Computing*, 30:1842–1862, 2001. [SICOMP:10.1137/S0097539700366528]. 4, 4.3

[3] * T. BATU, R. RUBINFELD, AND P. WHITE: Fast approximation PCPs for multidimensional bin-packing problems. *Information and Computation*, 196:42–56, 2005. [IandC:10.1016/j.ic.2004.10.001]. 2, 4

[4] * M. BELLARE, O. GOLDREICH, AND M. SUDAN: Free bits, PCPs, and nonapproximability – towards tight results. *SIAM Journal on Computing*, 27:804–915, 1998. [SICOMP:10.1137/S0097539700366528]. 2.2

[5] * E. BEN-SASSON, O. GOLDREICH, P. HARSHA, M. SUDAN, AND S. VADHAN: Robust PCPs of proximity, shorter PCPs and applications to coding. In *Proc.* 36^{th} *STOC*, pp. 1–10. ACM Press, 2004. [STOC:1007352.1007361]. 1, 2, 4, 4.2

[6] * E. BEN-SASSON, P. HARSHA, AND S. RASKHODNIKOVA: Some 3CNF properties are hard to test. *SIAM Journal on Computing*, 35:1–21, 2005. [SICOMP:10.1137/S0097539704445445]. 4

[7] * M. BLUM, M. LUBY, AND R. RUBINFELD: Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993. [JCSS:10.1016/0022-0000(93)90044-W]. 1, 1, 2.1

[8] * H. BUHRMAN, L. FORTNOW, I. NEWMAN, AND H. ROHRIG: Quantum property testing. In *Proc.* 14^{th} *ACM-SIAM Symp. on Discrete Algorithms (SODA'03)*, pp. 480–488. SIAM, 2003. [SODA:644108.644188]. 2, 2.3

[9] * F. ERGUN, S. KANNAN, R. KUMAR, R. RUBINFELD, AND M. VISWANATHAN: Spot checkers. *Journal of Computer and System Sciences*, 60:717–751, 2000. [JCSS:10.1006/jcss.1999.1692]. 1

[10] * F. ERGUN, R. KUMAR, AND R. RUBINFELD: Fast approximate probabilistically checkable proofs. *Information and Computation*, 189:135–159, 2004. [IandC:10.1016/j.ic.2003.09.005]. 1, 2, 4

[11] * E. FISCHER: The art of uninformed decisions: A primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001. 1

[12] * E. FISCHER AND L. FORTNOW: Tolerant versus intolerant testing for Boolean properties. In *Proc. 20th IEEE Conf. on Computational Complexity*, pp. 135–140. IEEE Computer Society Press, 2005. [CCC:10.1109/CCC.2005.30]. 1

[13] * E. FISCHER, G. KINDLER, D. RON, S. SAFRA, AND A. SAMORODNITSKY: Testing juntas. *Journal of Computer and System Sciences*, 68:753–787, 2004. [JCSS:10.1016/j.jcss.2003.11.004]. 2, 2

[14] * E. FISCHER AND I. NEWMAN: Testing versus estimation of graph properties. In *Proc.* 37^{th} *STOC*, pp. 138–146. ACM Press, 2005. [STOC:1060590.1060612]. 1

[15] * E. FISCHER, I. NEWMAN, AND J. SGALL: Functions that have read-twice constant width branching programs are not necessarily testable. *Random Structures and Algorithms*, 24:175–193, 2004. [RSA:10.1002/rsa.10110]. 4

[16] * O. GOLDREICH, S. GOLDWASSER, AND D. RON: Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998. [JACM:285055.285060]. 1

[17] * O. GOLDREICH AND L. TREVISAN: Three theorems regarding testing graph properties. *Random Structures and Algorithms*, 23:23–57, 2003. [RSA:10.1002/rsa.10078]. 1

[18] * J. HÅSTAD: Some optimal inapproximability results. *Journal of the ACM*, 48:798–859, 2001. [JACM:502090.502098]. 2

[19] * M. PARNAS, D. RON, AND R. RUBINFELD: Tolerant property testing and distance approximation. In *ECCC*, number TR04-010. 2004. [ECCC:TR04-010]. 1

[20] * M. PARNAS, D. RON, AND A. SAMORODNITSKY: Testing basic Boolean formulae. *SIAM Journal on Discrete Mathematics*, 16:20–46, 2002. [SIDMA:10.1137/S0895480101407444]. 2.2

[21] * D. RON: Property testing. In S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim, editors, *Handbook of Randomized Computing*, volume II, pp. 597–649. Kluwer, 2001. 1

[22] * R. RUBINFELD AND M. SUDAN: Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25:252–271, 1996. [SICOMP:10.1137/S0097539793255151]. 1

AUTHORS

Eldar Fischer
Faculty of Computer Science
Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel
eldar@cs.technion.ac.il
http://www.cs.technion.ac.il/users/eldar/

Lance Fortnow
Department of Computer Science
University of Chicago
1100 E. 58th St., Chicago, IL 60637, USA
fortnow@cs.uchicago.edu
http://people.cs.uchicago.edu/~fortnow/

ABOUT THE AUTHORS

ELDAR FISCHER completed his Ph. D. in 1999 at Tel-Aviv University under the supervision of Noga Alon and has been a member of the Faculty of Computer Science at the Israel Institute of Technology (the Technion) since 2001. His main interests lie in the analysis of algorithms, especially sublinear ones, but he is also known to dabble in Formal Logic and to investigate the odd combinatorial graph-theoretic question. His main extra-curricular interest centers on board and card games, especially the less well known ones.

LANCE FORTNOW received his Ph. D. under Michael Sipser in Applied Mathematics at MIT in 1989. He has spent his academic career at the University of Chicago with the exception of a senior research scientist postion at the NEC Research Institute from 1999 to 2003. In 1992 he received the NSF Presidential Faculty Fellowship and was a Fulbright Scholar visiting CWI in Amsterdam 1996-97. Fortnow studies computational complexity and its applications to electronic commerce, quantum computation, bioinformatics, learning theory and cryptography. His early work on interactive proofs precipitated the development of probabilistically checkable proofs and inapproximability theory. Fortnow writes the popular scientific and academic weblog Computational Complexity.