THEORY OF COMPUTING, Volume 7 (2011), pp. 155–176
www.theoryofcomputing.org

Distribution-Free Testing for Monomials with a Sublinear Number of Queries

Elya Dolev Dana Ron*

Received: November 6, 2010; published: September 8, 2011.

Abstract: We consider the problem of distribution-free testing of the class of monotone monomials and the class of monomials over n variables. While there are very efficient testers for a variety of classes of functions when the underlying distribution is uniform, designing distribution-free testers (which must work under an arbitrary and unknown distribution) tends to be more challenging. When the underlying distribution is uniform, Parnas et al. (*SIAM J. Discr. Math., 2002*) give a tester for (monotone) monomials whose query complexity does not depend on n, and whose dependence on the distance parameter is (inverse) linear. In contrast, Glasner and Servedio (*Theory of Computing, 2009*) prove that every distribution-free tester for monotone monomials as well as for general monomials must have query complexity $\widetilde{\Omega}(n^{1/5})$ (for a constant distance parameter ε).

In this paper we present distribution-free testers for these classes with query complexity $\widetilde{O}(n^{1/2}/\varepsilon)$. We note that in contrast to previous results for distribution-free testing, our testers do not build on the testers that work under the uniform distribution. Rather, we define and exploit certain structural properties of monomials (and functions that differ from them on a non-negligible part of the input space), which were not used in previous work on property testing.

ACM Classification: F.2.2, G.2.0, G.3

AMS Classification: 68Q25, 68W20, 68W25, 68W40

Key words and phrases: property testing, distribution-free testing, Boolean functions, monomials

*Research supported by the Israel Science Foundation (grant No. 246/08).

 DOI: 10.4086/toc.2011.v007a011

1 Introduction

Testers (for properties of functions) are algorithms that separate the functions with a prespecified property from the functions that are "far" from having the property with respect to some fixed distance measure. In most works on property testing, distance is measured with respect to the uniform distribution over the function domain. While in many contexts this distance is appropriate, as it corresponds to assigning equal "importance" or weight to each point in the domain, there are scenarios in which we may want to deal with an underlying weight distribution that is not uniform, and furthermore, is not known to the tester. We refer to the latter model as *distribution-free* property testing, while testing under the uniform distribution is considered to be the *standard* model. In the standard model the tester is given query access to the tested function; in the distribution-free model the tester is also given access to sample inputs distributed according to the unknown underlying distribution. By the *complexity* of the tester we mean its query complexity. (In the complexity of a tester we count both the queries on the points sampled according to the underlying distribution and the queries on points selected by the tester.)

Indeed, the notion of distribution-free testing is inspired by the distribution-free (Probably Approximately Correct (PAC)) learning model [19] and understanding the relation between testing and learning is one of the issues of interest in the study of property testing. As observed in [9], the complexity of testing a function class $\mathcal{F}$ (that is, testing the property of membership in $\mathcal{F}$) is not higher than (proper) learning the class $\mathcal{F}$ (under the same conditions, e. g., with respect to the uniform distribution or distribution-free). In view of this, a natural question is for what classes of functions is the complexity of testing *strictly* lower than that of learning. Note that, as opposed to learning, if we have a tester for (membership in) a class of functions $\mathcal{F}$, this does not imply that we have a tester (with similar complexity) for all subclasses $\mathcal{F}'$ of $\mathcal{F}$.

There is quite a large variety of function classes for which the complexity of testing is strictly lower than that of learning when the underlying distribution is uniform (e. g., linear functions [1], low-degree polynomials [17], singletons, monomials [16] and small monotone DNF [16], monotone functions (e. g., [5, 4]), small juntas [6], small decision lists, decision trees and (general) DNF [3] linear threshold functions [14], and more). In contrast, there are relatively few such positive results for distribution-free testing [10, 11, 12], and, in general, designing distribution-free testers tends to be more challenging.

One of the main positive results for distribution-free testing [10] is that every function class that has a standard tester and can be efficiently *self-corrected* [1], has a distribution-free tester whose complexity is similar to that of the standard tester. In particular this implies that there are efficient distribution-free testers for linear functions and more generally, for low-degree polynomials [10]. However, there are function classes of interest (in particular from the point of view of learning theory), which have efficient standard testers, but for which self-correctors do not exist (or are not known to exist). Several such classes (of Boolean functions over $\{0,1\}^n$) were studied by Glasner and Servedio [7]. Specifically, they consider monotone monomials, general monomials, decisions lists, and linear threshold functions. They prove that for these classes, in contrast to standard testing, where the query complexity does not depend on the number of variables n, every distribution-free tester must make $\Omega((n/\log n)^{1/5})$ queries (for a constant distance parameter ε). While these negative results establish that a strong dependence on n is unavoidable for these functions classes in the distribution-free case, it still leaves open the question of whether some *sublinear* dependence on n is possible (while distribution-free learning (with queries) requires at least linear query complexity [18]).

Our results In this work we prove that both for monotone monomials and for general monomials, a sublinear dependence on n can be achieved for distribution-free testing. Specifically, we describe distribution-free testers for these families whose query complexity is $O(\sqrt{n}\log n/\varepsilon)$. Thus we advance our knowledge concerning efficient distribution-free testing for two basic function classes. Furthermore, while previous distribution-free testers have been based on, and are similar to the corresponding standard testers, this is not the case for our testers. Rather, we define and exploit certain structural properties of monomials (and functions that differ from them in a non-negligible manner), which were not used in previous work on property testing in the standard model. In what follows we give some intuition concerning the difficulty encountered when trying to extend standard testing of (monotone) monomials to distribution-free testing and then shortly discuss the ideas behind our testers.

Standard vs. distribution-free testing of monomials The first simple observation concerning testing monomials under the uniform distribution is the following. If f is a k-monomial (that is, a conjunction of k literals), then $\Pr[f(x)=1]=2^{-k}$ (where the probability is over a uniformly selected x). This implies that we can effectively consider only relatively small monomials, that is, k-monomials for $k=\log(O(1/\varepsilon))$, and it allows the tester to have an exponential dependence on k (since this translates to a linear dependence on $1/\varepsilon$). This is not in general the case when the underlying distribution is arbitrary. In particular, the functions considered in the lower bound proof of [7] (some of which are monomials, and some of which are far from being monomials), depend on $\Omega(n)$ variables. Thus, for these functions, considering uniformly selected points, essentially gives no information (since the function assigns value 0 to all but a tiny fraction of the points). Furthermore, the support of the distribution D defined in [7] is such that the following holds. If one takes a sample (distributed according to D) of size smaller than the square-root of the support size of D (where there are roughly $n^{2/5}$ points in the support), and performs queries on the sampled points, then it is not possible to distinguish between the monomials and the functions that are far from being monomials (with respect to D). Thus, by sampling according to D, we essentially get no information unless the size of the sample is above a (fairly high) threshold. On the other hand, if we perform queries outside the support of D, then intuitively (and this is formalized in [7]), violations (with respect to being a monomial) are hard to find.

Before continuing with a high level description of our testers, we note that if we restrict the task of testing to distribution-free testing of (monotone) k-monomials, where k is fixed, then there is a tester whose query complexity grows exponentially with k. This follows by combining two results: (1) The aforementioned result of Halevy and Kushilevitz [10] concerning the use of "self-correction" in transforming standard testers to distribution-free testers; (2) The result of Parnas et al. [16] for testing (monotone) monomials, which has a self-corrector (with complexity 2^k) as a building block. This implies that for small k (i. e., $k \leq \log n - \omega(1)$), we have a tester with complexity that is sublinear in n. Hence, the question is what can be done when it is not assumed that k is small.

Our testers: Ideas and techniques In what follows we discuss the tester for monotone monomials over $\{0,1\}^n$. The tester for general monomials has the same high-level structure, and can be viewed as a generalization of the tester for monotone monomials.

Our tester tries to find evidence that the tested function f is not a monotone monomial (where if f is a monotone monomial then clearly the tester will not find such evidence). The tester looks for evidence in the form of a small subset of points $\{y^0, y^1, \ldots, y^t\}$, each in $\{0,1\}^n$, where $f(y^0) = 0$ and $f(y^i) = 1$ for each $1 \leq i \leq t$, such that *no* monotone monomial agrees with f on this subset.

Based on these subsets we introduce a notion that is central to our tester and its analysis: the *violation* hypergraph of f, which we denote by H_f. The vertex-set of H_f is $\{0,1\}^n$, and its edge-set corresponds to subsets of the form described above. By the definition of H_f, if f is a monotone monomial, then H_f has no edges. On the other hand, we prove that if f is far from being a monotone monomial (with respect to the underlying distribution, D), then every vertex cover of the (edges of) H_f must have relatively large weight (with respect to D). We use this fact to show that if f is far from being a monotone monomial, then our tester finds a (small) edge in H_f (with high probability). We next give a high level description of the tester.

Our tester works in two stages, where in each stage it takes a sample of size $O(\sqrt{n}/\varepsilon)$, distributed according to D. In the first stage it only considers the points y in the sample such that $y \in f^{-1}(0)$. If f is a monotone monomial, then for each such point y there must be at least one index j such that $y_j = 0$ and x_j is a variable in the monomial. Hence, for each point $y \in f^{-1}(0)$ that is selected in the first sample, the tester searches for such a *representative* index j (as explained in detail in Subsection 3.2). If any search fails, then the tester rejects, since it has evidence that f is not a monotone monomial. This evidence is in the form of an edge (of size 3) in H_f. Otherwise, the tester has a set J of indices.

In the second stage, the tester considers only the sample points $y \in f^{-1}(1)$. For each such sample point y it checks whether there exists an index $j \in J$ such that $y_j = 0$. If such an index exists, then an edge in H_f is found and the tester rejects. The crux of the proof is showing that if the probability that the tester does not find evidence (in both stages) is small, then it is possible to construct a small-weight vertex cover in H_f (implying that f is close to being a monotone monomial).

Other related work In addition to the results mentioned previously, Halevy and Kushilevitz [10] study distribution-free testing of monotonicity for functions $f : \Sigma^n \to R$ (where Σ and R are fully ordered). Building on the (one-dimensional) standard tester in [5] they give a distribution-free tester whose query complexity is $O((2\log|\Sigma|)^n/\varepsilon)$. Thus, the dependence on the dimension, n is exponential, in contrast to some of the standard testers for monotonicity [8, 4] where the dependence on n is linear.[1] Halevy and Kushilevitz [10] prove that the exponential dependence on n is unavoidable for distribution-free testing even in the case of Boolean functions over the Boolean hypercube (that is, $|\Sigma| = |R| = 2$). In [12], Halevy and Kushilevitz further the study of testing monotonicity over graph products.

Halevy and Kushilevitz [11] also study distribution-free testing of graph properties in sparse graphs, and give a distribution-free tester for connectivity, with similar complexity to the standard tester for this property.

We note that for some properties that have efficient standard testers, the testers can be extended to work under more general families of distributions such as product distributions (e. g., [6, 3]). In recent work, Kopparty and Saraf [13] consider tolerant testing [15] of linearity under non-uniform distributions (that have certain properties).

[1]To be precise, the complexity of the tester in [4] is $O(n\log|\Sigma|\log|R|/\varepsilon)$, where $|R|$ is the effective size of the range of the function, that is, the number of distinct values of the function.

Organization We start by introducing some notation and definitions in Section 2. In Section 3 we describe and analyze the distribution-free tester for monotone monomials, and in Section 4 we explain how to extend it to general monomials.

Further research Perhaps the first question that comes to mind is what is the exact complexity of distribution-free testing of (monotone) monomials given the gap between our upper bound and the lower bound of [7]. It will also be interesting to design sublinear testers for the other function classes studied in [7]. Another direction is to study testing of monomials and other basic function classes under known distributions (other than the uniform distribution).

2 Preliminaries

For an integer k we let $[k] \stackrel{\text{def}}{=} \{1,\ldots,k\}$. In all that follows we consider Boolean functions f whose domain is $\{0,1\}^n$.

Definition 2.1 (Monomials)**.** A function $f : \{0,1\}^n \to \{0,1\}$ is a *monomial* if it is a conjunction ("and") of a subset of the literals $\{x_1,\bar{x}_1,\ldots,x_n,\bar{x}_n\}$. It is a *monotone monomial* if it is a conjunction only of variables (and no negations of variables). We denote the class of monomials by $\mathcal{MON}$ and the class of monotone monomials by $\mathcal{MON}_{\mathcal{M}}$.

We note that we allow the special case that the subset of literals (variables) is empty, in which case f is the all-1 function. In Subsections 3.4 and 4.4 we discuss how to augment our tests so that they work for the case that the subset of literals (variables) must be non-empty.

Definition 2.2 (Distance)**.** Let D be a distribution over $\{0,1\}^n$. For two functions $f,g : \{0,1\}^n \to \{0,1\}$, we let

$$\mathrm{dist}_D(f,g) \stackrel{\text{def}}{=} \Pr_{x\sim D}[f(x) \neq g(x)] \tag{2.1}$$

denote the distance between f and g with respect to D. For a function $f : \{0,1\}^n \to \{0,1\}$ and a set of Boolean functions $\mathcal{F}_n$ over $\{0,1\}^n$, we let

$$\mathrm{dist}_D(f,\mathcal{F}_n) \stackrel{\text{def}}{=} \min_{g\in\mathcal{F}_n}\{\mathrm{dist}_D(f,g)\} \tag{2.2}$$

denote the distance between f and the set $\mathcal{F}_n$.

Note that the distance between f and $\mathcal{F}_n$ with respect to D can be 0 while $f \notin \mathcal{F}_n$.

Definition 2.3 (Distribution-Free Testing)**.** Let $\mathcal{F} = \{\mathcal{F}_n\}$ be a class of Boolean functions, where each $\mathcal{F}_n$ is a set of Boolean functions over $\{0,1\}^n$. A *distribution-free tester* for (membership in) $\mathcal{F}$ is given the value of n, access to examples that are distributed according to an unknown distribution D over $\{0,1\}^n$, and query access to an unknown function $f : \{0,1\}^n \to \{0,1\}$. The tester is also given a distance parameter $0 < \varepsilon < 1$, and is required to behave as follows.

- If $f \in \mathcal{F}_n$, then the tester should *accept* with probability at least $2/3$.

- If $\mathrm{dist}_D(f, \mathcal{F}_n) > \varepsilon$, then the tester should *reject* with probability at least $2/3$.

If the tester accepts every $f \in \mathcal{F}_n$ with probability 1, then it is a *one-sided error* tester.

In all that follows f always denotes the (unknown) tested function, and D denotes the (unknown) underlying distribution with respect to which the tester should work. For a point $y \in \{0,1\}^n$ let $D(y)$ denote the probability assigned to y by D, and for a subset $S \subseteq \{0,1\}^n$ let $D(S) = \sum_{y \in S} D(y)$ denote the weight that D assigns to the subset S. For the sake of simplicity, with a slight abuse of notation, we may write $f \in \mathcal{F}$ (instead of $f \in \mathcal{F}_n$), and $\mathrm{dist}_D(f, \mathcal{F})$ (instead of $\mathrm{dist}_D(f, \mathcal{F}_n)$).

As noted in the introduction, by the complexity of a tester we mean its query complexity (which includes both queries on examples that are generated according to the unknown underlying distribution D as well as queries on points selected by the tester).

We assume without loss of generality that $\varepsilon \geq 2^{-n}$ since otherwise, by performing a number of queries that is linear in $1/\varepsilon$ (that is, querying f on all domain elements), it is possible to determine whether or not f is a monotone monomial.

3 Distribution-free testing of monotone monomials

We start by introducing the notion of a *violation hypergraph* of a function and establishing its relation to (the distance to) monotone monomials.

3.1 The violation hypergraph

Before defining the violation hypergraph, we introduce some notation. For each point $y \in \{0,1\}^n$ let

$$Z(y) \stackrel{\text{def}}{=} \{j : y_j = 0\} \quad \text{and let} \quad O(y) \stackrel{\text{def}}{=} \{j : y_j = 1\}\,.$$

We use 1^n to denote the all-1 vector (point).

Let g be a Boolean function over $\{0,1\}^n$ and let $\{y^0, y^1, \ldots, y^t\} \subseteq \{0,1\}^n$ be a subset of points such that $g(y^0) = 0$ and $g(y^i) = 1$ for all $1 \leq i \leq t$. A simple but useful observation is that if g is a monotone monomial, then $Z(y^0)$ must include at least one index j such that $j \in \bigcap_{i=1}^t O(y^i)$. This is true because if g is a monotone monomial, then the subset of indices that correspond to the variables that g is a conjunction of must be a subset of $\bigcap_{i=1}^t O(y^i)$. But then there must be at least one index $j \in \bigcap_{i=1}^t O(y^i)$ for which $y^0_j = 0$, or else $g(y^0)$ would be 1. In other words, if $Z(y^0)$ does not include any index j such that $j \in \bigcap_{i=1}^t O(y^i)$, which is equivalent to $\bigcap_{i=1}^t O(y^i) \subseteq O(y^0)$, then g cannot be a monotone monomial. This observation motivates the next definition.

Definition 3.1 (Violation Hypergraph of f)**.** Let $H_f = (V(H_f)), E(H_f))$ be the hypergraph whose vertex set, $V(H_f)$ is $\{0,1\}^n$, and whose edge set, $E(H_f)$, contains all subsets $\{y^0, y^1, \ldots, y^t\} \subseteq \{0,1\}^n$, $t \geq 1$, of the following form:

- $f(y^0) = 0$ and $f(y^i) = 1$ for all $1 \leq i \leq t$.
- $\bigcap_{i=1}^t O(y^i) \subseteq O(y^0)$.

If $f(1^n) = 0$, then $\{1^n\} \in E(H_f)$ as well.

For example, suppose that $f(0011) = 0$, $f(0110) = 1$ and $f(1011) = 1$. Then $O(0011) = \{3,4\}$, $O(0110) = \{2,3\}$, and $O(1011) = \{1,3,4\}$, so that $O(0110) \cap O(1011) = \{3\}$, which is a subset of $O(0011)$, implying that $\{0011, 0110, 1011\}$ is an edge in H_f. Also note that the edge-set of the hypergraph may be exponentially large in n, and edges may have large size (e. g., $\Omega(n)$). Finally, observe that the second condition in the definition of the edges of H_f, that is, $\bigcap_{i=1}^t O(y^i) \subseteq O(y^0)$, is equivalent to $Z(y^0) \subseteq \bigcup_{i=1}^t Z(y^i)$.

By the observation preceding Definition 3.1, if f is a monotone monomial, then $E(H_f) = \emptyset$. We next claim that the reverse implication holds as well, so that we obtain a characterization of monotone monomials that is based on H_f.

Lemma 3.2. *If $E(H_f) = \emptyset$, then f is a monotone monomial.*

Lemma 3.2 follows directly from the next claim (setting $R = \{0,1\}^n$). The claim will also serve us in proving an additional lemma.

Claim 3.3. *Let R be a subset of $\{0,1\}^n$ and let $H_f(R) = (R, E(H_f(R)))$ be the sub-hypergraph of H_f where $E(H_f(R))$ consists of all edges in H_f that are contained[2] in R. If $E(H_f(R)) = \emptyset$ then there exists $h \in \mathcal{MON}_{\mathcal{M}}$ such that $h(x) = f(x)$ for every $x \in R$.*

Proof. If $f^{-1}(1) \cap R = \emptyset$, that is, $R \subseteq f^{-1}(0)$, then, since $E(H_f(R)) = \emptyset$, necessarily $1^n \notin R$. In this case we let h be the monomial that is the conjunction of all variables x_j (so that it has value 1 only on 1^n and 0 elsewhere). Since $f^{-1}(1) \cap R = \emptyset$, this monomial is consistent with all points in R.

Otherwise, let

$$M = \bigcap_{y \in f^{-1}(1) \cap R} O(y) .$$

Note that since $E(H_f(R)) = \emptyset$, necessarily $f(1^n) = 1$, and so $f^{-1}(1) \neq \emptyset$. Let h be the monotone monomial that is the conjunction of all x_j such that $j \in M$ (where if M is empty, then h is the all-1 function). That is, $h(x) = \bigwedge_{j \in M} x_j$. We next show that $f(y) = h(y)$ for all $y \in \{0,1\}^n \cap R$.

By the definition of h, for all $y \in f^{-1}(1) \cap R$, $h(y) = 1$. To establish that $h(y) = 0$ for all $y \in f^{-1}(0) \cap R$, assume in contradiction that there is a point $y^0 \in f^{-1}(0) \cap R$ such that $h(y^0) = 1$. Since $h(y^0) = 1$ we have (by the definition of h) that $y^0_j = 1$ for all $j \in M$. That is,

$$\bigcap_{y \in f^{-1}(1) \cap R} O(y) \subseteq O(y^0) .$$

But this means that $\{y^0\} \cup (f^{-1}(1) \cap R)$ is an edge in $H_f(R)$ and we have reached a contradiction. □

Recall that a vertex cover of a hypergraph is a subset of the vertices that intersects every edge in the hypergraph. We next establish that if f is far from being a monotone monomial (with respect to D), then every vertex cover of H_f must have large weight (with respect to D). This lemma strengthens Lemma 3.2

[2] We could refer to this as the sub-hypergraph *induced* by R, but this term is usually used for a slightly different notion in the context of hypergraphs.

in the following sense. Lemma 3.2 is equivalent to saying that if f is not a monotone monomial, then $E(H_f) \neq \emptyset$. In particular this implies that if f is not a monotone monomial, then every vertex cover of H_f is non-empty. Lemma 3.4 can be viewed as quantifying this statement (and taking into account the underlying distribution D).

Lemma 3.4. *If $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) > \varepsilon$, then for every vertex cover C of H_f we have $D(C) > \varepsilon$.*

Proof. Assume, contrary to the claim, that there exists a vertex cover C of H_f such that $D(C) \leq \varepsilon$. Let $R = \{0,1\}^n \setminus C$ be the vertices that do not belong to the vertex-cover. Since C is a vertex cover, we have that $E(H_f(R)) = \emptyset$. By Claim 3.3 there is a monotone monomial that is consistent with f on R. Since $D(C) \leq \varepsilon$ this implies that $\mathrm{dist}_D(h, f) \leq \varepsilon$, in contradiction to the premise of the lemma. □

By Lemmas 3.2 and 3.4, if f is a monotone monomial, then $E(H_f) = \emptyset$, so that trivially every minimum vertex cover of H_f is empty, while if $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) > \varepsilon$, then every vertex cover of H_f has weight greater than ε with respect to D. We would like to show that this implies that if $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) > \varepsilon$, then we can actually find (with high probability) an edge in H_f, which provides evidence to the fact that f is not a monotone monomial.

3.2 The tester

We first introduce some more notation. Let $\bar{e}^j = 1^{j-1}01^{n-j}$. For any subset $Z \subseteq [n]$, let $y(Z)$ be the point in $\{0,1\}^n$ such that for every $j \in Z$ its j^{th} coordinate is 0, and for every $j \notin Z$ its j^{th} coordinate is 1. For any subset $S \subseteq \{0,1\}^n$, let $S_{f,0} = \{y \in S : f(y) = 0\}$ and $S_{f,1} = \{y \in S : f(y) = 1\}$.

The first observation on which our tester is based is that for every point $y \in f^{-1}(0)$, there must be at least one index $j \in Z(y)$ for which $f(\bar{e}^j) = 0$, or else we have evidence that f is not a monotone monomial. In fact, we do not need to verify that $f(\bar{e}^j) \neq 0$ for *every* $j \in Z(y)$ in order to obtain evidence that f is not a monotone monomial. Rather, if we search for such an index (in a manner described momentarily), and this search fails, then we already have evidence that f is not a monotone monomial.

The search procedure (which performs a binary search), receives as input a point $y \in f^{-1}(0)$ and searches for an index $j \in Z(y)$ such that $f(\bar{e}^j) = 0$. This is done by repeatedly partitioning a set of indices, Z, starting with $Z = Z(y)$, into two parts Z_1 and Z_2 of (almost) equal size, and continuing the search with a part Z_i, $i \in \{1,2\}$ for which $f(y(Z_i)) = 0$. (If both parts satisfy the condition, then we continue with Z_1.) Note that if both $f(y(Z_1)) = 1$ and $f(y(Z_2)) = 1$, then we have evidence that f is not a monotone monomial because $f(y(Z_1 \cup Z_2)) = 0$ (so that $\{y(Z_1 \cup Z_2), y(Z_1), y(Z_2)\}$ is an edge in H_f). The search also fails (from the start) if $Z(y) = \emptyset$ (that is, $y = 1^n$). For the precise pseudo-code of the procedure, see Figure 1.

The tester starts by obtaining a sample of $\Theta(\sqrt{n}/\varepsilon)$ points, where each point is generated independently according to D. (Since the points are generated independently, repetitions may occur.) For each point in the sample that belongs to $f^{-1}(0)$, the tester calls the binary search procedure. If any search fails, then the tester rejects f (recall that in such a case the tester has evidence that f is not a monotone monomial). Otherwise, the tester has a collection of indices J such that $f(\bar{e}^j) = 0$ for every $j \in J$. The tester then takes an additional sample, also of size $\Theta(\sqrt{n}/\varepsilon)$, and checks whether there exists a point y in the sample such that $f(y) = 1$ and $Z(y)$ contains some $j \in J$. In such a case the tester has evidence

Algorithm 1. Binary Search (Input: $y \in \{0,1\}^n$)

1. $Z \leftarrow Z(y)$.
2. if $|Z| = 0$, then output *fail* and halt.
3. While $(|Z| \geq 2)$ do

   Let (Z_1, Z_2) be a fixed partition of Z where $||Z_1| - |Z_2|| \leq 1$.
   Specifically, Z_1 is the set of the first $\lfloor |Z|/2 \rfloor$ indices in Z.
   - If $f(y(Z_1)) = 0$, then $Z \leftarrow Z_1$;
   - else if $f(y(Z_2)) = 0$, then $Z \leftarrow Z_2$;
   - else output *fail* and halt.
4. Output the single index that remains in Z.

Figure 1: The binary search procedure for monotone monomials.

that f is not a monotone monomial (specifically, $\{\bar{e}^j, y\}$ is an edge in H_f), and it rejects. For the precise pseudo-code of the tester, see Figure 2.

We shall use Lemma 3.4 to show that if $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}})$ is relatively large, then either the first sample will contain a point on which the binary search procedure fails (with high probability over the choice of the first sample), or the second sample will contain a point y such that $f(y) = 1$ and $Z(y) \cap J \neq \emptyset$ (with high probability over the choice of both samples).

Algorithm 2. Monotone Monomials Test

1. Obtain a sample T of $\Theta(\sqrt{n}/\varepsilon)$ points, each generated independently according to D.
2. For each point $y \in T_{f,0}$ run the binary search procedure (Algorithm 1) on y.
3. If the binary search fails for any of the points, then output *reject* and halt. Otherwise, for each $y \in T_{f,0}$ let $j(y)$ be the index returned for y, and let $J(T_{f,0}) = \bigcup_{y \in T_{f,0}} \{j(y)\}$.
4. Obtain another sample T' of size $\Theta(\sqrt{n}/\varepsilon)$ (generated independently according to D).
5. If there is a point $y \in T'_{f,1}$ such that $Z(y) \cap J(T_{f,0}) \neq \emptyset$, then output *reject*, otherwise output *accept*.

Figure 2: The distribution-free tester for monotone monomials.

3.3 The analysis of the tester for monotone monomials

The next definition will serve us in the analysis of the tester.

Definition 3.5 (Empty points and representative indices)**.** For a point $y \in f^{-1}(0)$, we say that y is *empty* (with respect to f) if the binary search procedure (Algorithm 1) fails on y. We denote the set of empty points (with respect to f) by $Y_\emptyset(f)$. If y is not empty, then we let $j(y) \in Z(y)$ denote the index that the binary search procedure returns. We refer to this index as the *representative* index for y. If $y \in Y_\emptyset(f)$, then $j(y)$ is defined to be 0.

Note that since the binary search procedure is deterministic, the index $j(y)$ is uniquely defined for each $y \notin Y_\emptyset(f)$.

As in Algorithm 2, for a sample T and $T_{f,0} = T \cap f^{-1}(0)$, we let

$$J(T_{f,0}) = \{j(y) : y \in T_{f,0} \setminus Y_\emptyset(f)\}$$

denote the set of representative indices for the sample. For any subset $J \subseteq [n]$, let $Y_{f,1}(J)$ denote the set of all points $y \in f^{-1}(1)$ for which $Z(y) \bigcap J \neq \emptyset$. In particular, if we set $J = J(T_{f,0})$, then each point $y \in Y_{f,1}(J)$, together with any index j in the intersection of $Z(y)$ with J, provide evidence that f is not a monotone monomial (i. e., $\{\bar{e}^j, y\} \in E(H_f)\}$). We next state our main lemma.

Lemma 3.6. *Suppose that $dist_D(f, \mathcal{MON}_\mathcal{M}) > \varepsilon$ and consider a sample T of $c_1\sqrt{n}/\varepsilon$ points generated independently according to D. For a sufficiently large constant c_1, with probability at least $5/6$ over the choice of T, either $T_{f,0}$ contains an empty point (with respect to f) or $D(Y_{f,1}(J(T_{f,0}))) \geq \varepsilon/(4\sqrt{n})$.*

As we show subsequently, the correctness of the tester follows quite easily from Lemma 3.6. Before proving Lemma 3.6 in detail, we give the high level idea of the proof. Lemma 3.6 is established by proving the contrapositive statement. Namely, if the probability (over the choice of T) that $T_{f,0}$ does not contain an empty point (with respect to f) and $D(Y_{f,1}(J(T_{f,0}))) < \varepsilon/(4\sqrt{n})$ is at least $1/6$, then $\mathrm{dist}_D(f, \mathcal{MON}_\mathcal{M}) \leq \varepsilon$. This is proved by applying a probabilistic argument to construct a vertex cover C of H_f such that $D(C) \leq \varepsilon$ (assuming the counter-assumption holds), and then applying (the contrapositve of) Lemma 3.4.

Specifically, we first put in the cover C all empty points. This takes care of all edges that contain empty points, where by the (first part of) the counter-assumption, the total weight of all empty points is very small. In the next stage we work in $O(\sqrt{n})$ iterations. Each iteration ℓ (except, possibly, for the last iteration), is associated with a subset J^ℓ of new representative indices (i. e., indices that were not associated with previous iterations). We prove that in each iteration (except, possibly, the last) there exists such a subset of indices having size $\Omega(\sqrt{n})$. The subset of points (all from $f^{-1}(1)$) that are added to C in iteration ℓ covers all edges $\{y^0, y^1, \ldots, y^t\}$ such that $j(y^0) \in J^\ell$. The second part of the counter-assumption is used to ensure that the weight of each subset of points that is added to the cover is $O(\varepsilon/\sqrt{n})$ (so that we get a total weight of $O(\varepsilon)$ over all iterations). In the last stage we add to C all points in $f^{-1}(0)$ that reside in edges that are not yet covered, where we can show that the total weight of these points is $O(\varepsilon)$ as well.

The above discussion hints to the reason why the sizes of the samples taken by the tester grow like $\sqrt{n}$. Roughly speaking, if the first sample, T, is significantly smaller than $\sqrt{n}$, then the second sample, T', has to be significantly larger. In other words, if we want to decrease the size of the sample T in Lemma 3.6, then we also need to decrease the lower bound on $D(Y_{f,1}(J(T_{f,0})))$. The reason is that as the size of T decreases, the sizes of the subsets J^ℓ (defined in the proof of Lemma 3.6) decrease as well, and the number of iterations in the construction of the vertex cover C increases. But then, in order to obtain a vertex cover with small total weight, the weights of the subsets of points that are added in each iteration, must be smaller.

More formally, the proof of Lemma 3.6 builds on Claim 3.7, stated next, which in turn uses the following notation: For a subset $J \subseteq [n]$, we let $Y_{f,0}(J)$ denote the set of points $y \in f^{-1}(0)$ for which[3] $j(y) \in J$.

Claim 3.7. *Let I be any fixed subset of $[n]$, and consider a sample T of $s = c_1\sqrt{n}/\varepsilon$ points generated independently according to D. For a sufficiently large constant c_1, with probability at least $9/10$ over the choice of T, either*

$$|J(T_{f,0}) \setminus I| \geq \sqrt{n} \quad or \quad D(Y_{f,0}([n] \setminus (J(T_{f,0}) \cup I))) \leq \varepsilon/2\,.$$

To make Claim 3.7 more concrete, consider the special case in which $I = \emptyset$. The lemma simply says that with probability at least $9/10$ (over the choice of T) either the subset of indices $J(T_{f,0})$ is relatively large, or the weight of the set of points y in $f^{-1}(0)$ for which $j(y)$ is not contained in $J(T_{f,0})$ is relatively small.

Proof. Consider selecting the points in T one after the other, where for $\ell = 1,\dots,s = c_1\sqrt{n}/\varepsilon$ we let y^ℓ denote the ℓ^{th} point selected, and we let $T^\ell = \{y^1,\dots,y^\ell\}$ (where $T^0 = \emptyset$). Let

$$Y^\ell = Y_{f,0}([n] \setminus (J(T^{\ell-1}_{f,0}) \cup I))$$

denote the set of points in $f^{-1}(0)$ whose representative index differs from all the previously found representative indices and does not belong to I. For each $1 \leq \ell \leq s$, let χ^ℓ be a $0/1$-valued random variable, where $\chi^\ell = 1$ if either $D(Y^\ell) \geq \varepsilon/2$ and $y^\ell \in Y^\ell$ (so that $j(y^\ell)$ is a *new* representative index), or $D(Y^\ell) < \varepsilon/2$. Thus, in either case $\Pr[\chi^\ell = 1] \geq \varepsilon/2$. Observe that if for some ℓ_0 we have that $D(Y^{\ell_0}) < \varepsilon/2$, then $D(Y^\ell) < \varepsilon/2$ for all $\ell \geq \ell_0$.

While $\chi^1,\dots,\chi^s$ are not independent random variables, we can bound the probability that their sum is smaller than $\sqrt{n}$ by considering slightly different random variables, which are independent.[4] For each $1 \leq \ell \leq s$, let $p_\ell = \Pr[\chi^\ell = 1]$, so that if $D(Y^\ell) \geq \varepsilon/2$, then $p_\ell = D(Y^\ell)$, and if $D(Y^\ell) < \varepsilon/2$, then $p_\ell = 1$. Let $\rho^1,\dots,\rho^s$ be $0/1$-valued random variable, where ρ^ℓ is the outcome of a coin-flip having bias $\varepsilon/(2p_\ell)$. Finally, let $\tilde{\chi}^1,\dots,\tilde{\chi}^s$ be random variables that are defined as follows based on $\chi^1,\dots,\chi^s$ and $\rho^1,\dots,\rho^s$: $\tilde{\chi}^\ell = 1$ if $\chi^\ell = 1$ *and* $\rho^\ell = 1$. The main two observations concerning each $\tilde{\chi}^\ell$ are:

1. For *any* outcome of $\tilde{\chi}^1,\dots,\tilde{\chi}^{\ell-1}$ we have that $\Pr[\tilde{\chi}^\ell = 1 \mid \tilde{\chi}^1,\dots,\tilde{\chi}^{\ell-1}] = \varepsilon/2$. Thus, $\tilde{\chi}^1,\dots,\tilde{\chi}^s$ are independent, equally distributed, random variables.

2. If $\tilde{\chi}^\ell = 1$, then necessarily $\chi^\ell = 1$. Therefore, for any threshold t we have that

$$\Pr\left[\sum_{\ell=1}^{s} \chi^\ell \geq t\right] \geq \Pr\left[\sum_{\ell=1}^{s} \tilde{\chi}^\ell \geq t\right].$$

[3]Note that $Y_{f,0}(J)$ and $Y_{f,1}(J)$ do not only differ in that they refer to points in $f^{-1}(0)$ and $f^{-1}(1)$, respectively. Rather, $Y_{f,0}(J)$ is the subset of points y (in $f^{-1}(0)$) whose *representative* index $j(y)$ belongs to J, while $Y_{f,1}(J)$ is the subset of points y (in $f^{-1}(1)$) for which *some* index $j \in Z(y)$ belongs to J.

[4]We note that we could apply an analysis using martingales (and Azuma's inequality), but then we would get a higher dependence on $1/\varepsilon$.

It therefore suffices to analyze the behavior of $\tilde{\chi}^1, \ldots, \tilde{\chi}^s$. Letting $c_1 = 64$ (so that $s = 64\sqrt{n}/\varepsilon$), by the foregoing discussion and by applying a multiplicative Chernoff bound [2], we have that

$$\begin{aligned} \Pr\left[\sum_{\ell=1}^{s} \chi^\ell < \sqrt{n}\right] &\leq \Pr\left[\sum_{\ell=1}^{s} \tilde{\chi}^\ell < \sqrt{n}\right] \\ &< \Pr\left[\sum_{\ell=1}^{s} \tilde{\chi}^\ell < (1/2)(\varepsilon/2)s\right] \\ &< \exp(-\varepsilon s/16) < 1/10 . \end{aligned} \tag{3.1}$$

This means that with probability at least $9/10$, either $D(Y^\ell) < \varepsilon/2$ for some ℓ, in which case we have $D(Y_{f,0}([n] \setminus (J(T_{f,0}) \cup I))) < \varepsilon/2$, or $|J(T_{f,0}) \setminus I| \geq \sqrt{n}$. □

Proof of Lemma 3.6. Assume, contrary to the claim, that with probability at least $1/6$ over the choice of T (which consist of $c_1\sqrt{n}/\varepsilon$ points selected independently according to D), there is no point in T that is empty with respect to f and $D(Y_{f,1}(J(T_{f,0}))) < \varepsilon/(4\sqrt{n})$. We will show how, using this assumption, it is possible to prove that there exists a vertex cover C of H_f with $D(C) \leq \varepsilon$. By Claim 3.4, this contradicts the fact that $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) > \varepsilon$.

THE CONSTRUCTION OF C. We show how to construct a vertex cover C of H_f in three stages. In the first stage we put in C all empty points, that is, all points in $Y_\emptyset(f)$. By the counter-assumption, $D(Y_\emptyset(f)) \leq 2\varepsilon/(c_1\sqrt{n})$. This is true because otherwise, the probability that the sample T does not contain an empty point is at most

$$\left(1 - \frac{2\varepsilon}{c_1\sqrt{n}}\right)^{c_1\sqrt{n}/\varepsilon} < \mathrm{e}^{-2} < 1/6 .$$

In the second stage we work in iterations. In each iteration ℓ we add to the cover a subset of points $Y^\ell \subseteq f^{-1}(1)$, which is determined by a subset $T^\ell \subseteq \{0,1\}^n$. These subsets are determined as follows. Let $I^1 = \emptyset$, and for $\ell > 1$ let $I^\ell = I^{\ell-1} \cup J(T_{f,0}^{\ell-1})$. We shall think of I^ℓ as the subset of representative indices that have "already been covered" in a sense that will become clear momentarily. Suppose we apply Claim 3.7 with $I = I^\ell$. The claim says that with probability at least $9/10$ over the choice of T, either $|J(T_{f,0}) \setminus I^\ell| \geq \sqrt{n}$ (that is, there are many "new" representative indices corresponding to points in $T_{f,0}$), or $D(Y_{f,0}([n] \setminus (J(T_{f,0}) \cup I^\ell))) \leq \varepsilon/2$ (that is, the total weight of the points whose representative index is not already in I^ℓ or $J(T_{f,0})$ is small). Thus, the probability that neither of the two hold is at most $1/10$.

Combining this with our counter-assumption (and taking a union bound), we have that with probability at least $1/6 - 1/10 > 0$ over the choice of T: $D(Y_{f,1}(J(T_{f,0}))) < \frac{\varepsilon}{4\sqrt{n}}$ (that is, the total weight of the points y in $f^{-1}(1)$ such that $Z(y) \cap J(T_{f,0}) \neq \emptyset$ is small), and either $|J(T_{f,0}) \setminus I^\ell| \geq \sqrt{n}$ or $D(Y_{f,0}([n] \setminus (J(T_{f,0}) \cup I^\ell))) \leq \varepsilon/2$. Since this (combined) event occurs with probability greater than 0, there must exist at least one set T for which it holds. Denote this set by T^ℓ, and let $Y^\ell = Y_{f,1}(J(T_{f,0}^\ell))$ (so that all points in Y^ℓ are added to the cover). If $D(Y_{f,0}([n] \setminus (J(T_{f,0}^\ell) \cup I^\ell))) \leq \varepsilon/2$, then this (second) stage ends, and in the third stage we add to the cover C all points in $Y_{f,0}([n] \setminus (J(T_{f,0}^\ell) \cup I^\ell))$. Otherwise, we set $I^{\ell+1} = I^\ell \cup J(T_{f,0}^\ell)$ and continue with the next iteration.

ESTABLISHING THAT C IS A VERTEX COVER OF H_f. Consider any point $y \in f^{-1}(0)$ that is contained in at least one edge in H_f. We shall show that either $y \in C$, or, for every edge $B \in H_f$ that contains y, there is at least one point $y' \in B \cap f^{-1}(1)$ that belongs to C. Since each edge in H_f contains some point $y \in f^{-1}(0)$, this implies that C is a vertex cover. Details follow.

If $y \in Y_\emptyset(f)$, then it is added to the cover in the first stage. In particular, if $f(1^n) = 0$ (so that H_f contains the edge $\{1^n\}$), then $1^n \in Y_\emptyset(f)$, implying that the edge $\{1^n\}$ is covered. Otherwise ($y \notin Y_\emptyset(f)$), we have that y has a representative index $j(y) \in Z(y)$. Consider the iterations in the second stage of the construction of C. If for some iteration ℓ we have that $j(y) \in J(T^\ell_{f,0})$ (where we consider the first iteration in which this occurs), then the cover contains all points in $Y_{f,1}(\{j(y)\}) \subseteq Y_{f,1}(J(T^\ell_{f,0})) = Y^\ell$. Since, by the definition of H_f, each edge in H_f that contains y must contain some $y' \in f^{-1}(1)$ such that $j(y) \in Z(y')$, all edges containing y are covered after iteration ℓ. On the other hand, if $j(y) \notin J(T^\ell_{f,0})$ for every iteration ℓ, then let ℓ^* denote the index of the last iteration and consider the third stage. In this stage we add to C all points in $Y_{f,0}([n] \setminus (J(T^{\ell^*}_{f,0}) \cup I^{\ell^*}))$. But this set consists of those points whose representative index is not contained in $J(T^{\ell^*}_{f,0}) \cup I^{\ell^*} = \bigcup_{\ell=1}^{\ell^*} J(T^\ell_{f,0})$, and in particular the set contains y, so that y is added to the cover in the third stage.

BOUNDING THE WEIGHT OF C. The main observation is that in each iteration ℓ of the second stage except, possibly, for the last one (ℓ^*), we have that $|J(T^\ell_{f,0}) \setminus I^\ell| \geq \sqrt{n}$. That is, each such iteration corresponds to a subset of at least $\sqrt{n}$ indices in $[n]$, where the different subsets are disjoint. This implies that $\ell^* \leq \sqrt{n}+1$. By the construction of C,

$$\begin{aligned} D(C) &= D(Y_\emptyset(f)) + \sum_{\ell=1}^{\ell^*} D\left(Y_{f,1}(J(T^\ell_{f,0}))\right) + D\left(Y_{f,0}([n] \setminus (J(T^{\ell^*}_{f,0}) \cup I^{\ell^*}))\right) \\ &\leq \frac{2\varepsilon}{c_1\sqrt{n}} + (\sqrt{n}+1)\cdot\frac{\varepsilon}{4\sqrt{n}} + \varepsilon/2 \;\leq\; \varepsilon\,, \end{aligned} \tag{3.2}$$

where the last inequality holds for $c_1 \geq 8$ (assuming $n \geq 4$). □

Theorem 3.8. *Algorithm 2 is a distribution-free 1-sided-error tester for (membership in) $\mathcal{MON}_\mathcal{M}$. Its query complexity is $O(\sqrt{n}\log n/\varepsilon)$.*

Proof. Consider first the case that f is a monotone monomial. Observe that the tester rejects only if it finds evidence that f is not a monotone monomial. This evidence is either in the form of two (disjoint) subsets of indices, Z_1 and Z_2 such that $f(y(Z_1)) = f(y(Z_2)) = 1$ while $f(y(Z_1 \cup Z_2))) = 0$ (found by the binary search procedure), or it is of the form of an index j and a point $y \in f^{-1}(1)$, such that $f(\bar{e}^j) = 0$ and $j \in Z(y)$. Therefore, the tester never rejects a monotone monomial.

Consider next the case that $\mathrm{dist}_D(f, \mathcal{MON}_\mathcal{M}) > \varepsilon$. By Lemma 3.6, for a sufficiently large constant c_1 in the $\Theta(\cdot)$ notation for T (the first sample), with probability at least $5/6$ over the choice of T, either there is an empty point in $T_{f,0} \subseteq T$, or $D(Y_{f,1}(J(T_{f,0}))) \geq \varepsilon/(4\sqrt{n})$. If there is an empty point in $T_{f,0}$, then the binary search will fail on that point and the tester will reject. On the other hand, if $D(Y_{f,1}(J(T_{f,0}))) \geq \varepsilon/(4\sqrt{n})$, then, since the size of the second sample, T', is $c_1'\sqrt{n}/\varepsilon$, the probability that no point $y \in Y_{f,1}(J(T_{f,0}))$ is selected in T' is at most $(1-\varepsilon/(4\sqrt{n}))^{c_1'\sqrt{n}/\varepsilon}$, which is upper bounded by $1/6$ for $c'_1 \geq 8$. But if such a point is selected, then the tester rejects.[5] Therefore, the probability that the tester rejects a function f for which $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) > \varepsilon$ is at least $2/3$.

Finally, the number of points sampled is $O(\sqrt{n}/\varepsilon)$ since the tester obtains two samples of this size. Since for each point in the first sample that belongs to $f^{-1}(0)$ the tester performs a binary search, the query complexity of the tester is $O(\sqrt{n}\log n/\varepsilon)$. □

3.4 Disallowing the all-1 function

Our tester as described in Subsection 3.2 works for the class of monotone monomials, $\mathcal{MON}_{\mathcal{M}}$, when it is assumed to contain the "empty" monomial, that is, the all-1 function. Let $\mathbf{1}$ denote the all-1 function (over $\{0,1\}^n$) and let $\mathcal{MON}'_{\mathcal{M}}$ denote the class of monotone monomials that are a conjunction of at least one variable. Thus, $\mathcal{MON}'_{\mathcal{M}} = \mathcal{MON}_{\mathcal{M}} \setminus \{\mathbf{1}\}$. We next show how to augment our tester for $\mathcal{MON}_{\mathcal{M}}$ (Algorithm 2) so that it work for $\mathcal{MON}'_{\mathcal{M}}$.

First we run Algorithm 2 with the distance parameter ε set to $\varepsilon/2$ and with slightly larger constants in the sizes of the sample, so that its failure probability is at most $1/6$ rather than $1/3$. If the tester rejects, then we reject as well. Otherwise, it is possible that the tester accepted f because $\mathrm{dist}_D(f,\mathbf{1}) \leq \varepsilon/2$ (so that $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) \leq \varepsilon/2$) while $\mathrm{dist}_D(f, \mathcal{MON}'_{\mathcal{M}}) > \varepsilon$. We are interested in detecting this (with high probability) and rejecting f in such a case.

To this end we take an additional sample R of $\Theta(\log n/\varepsilon)$ points, each generated independently according to D. We then check whether there exists at least one index $j \in [n]$ such that $y_j = 1$ for all points $y \in R \cap f^{-1}(1)$. If there is no such index, then we reject, otherwise we accept. We next show that if $\mathrm{dist}_D(f,\mathbf{1}) \leq \varepsilon/2$ and $\mathrm{dist}_D(f, \mathcal{MON}'_{\mathcal{M}}) > \varepsilon$, then this simple procedure rejects f with high constant probability.

The simple observation is that if $\mathrm{dist}_D(f,\mathbf{1}) \leq \varepsilon/2$ and $\mathrm{dist}_D(f, \mathcal{MON}'_{\mathcal{M}}) > \varepsilon$, then $\Pr_{y\sim D}[f(y) = 1 \text{ and } y_j = 0] \geq \varepsilon/2$ for every $j \in [n]$. Assume, contrary to the claim, that there exists some index $j \in [n]$ for which

$$\Pr_{y\sim D}[f(y) = 1 \text{ and } y_j = 0] < \varepsilon/2\,.$$

Since $\mathrm{dist}_D(f,\mathbf{1}) \leq \varepsilon/2$ (so that $\Pr_{y\sim D}[f(y) = 0] \leq \varepsilon/2$), we have that

$$\Pr_{y\sim D}[f(y) = 0 \text{ and } y_j = 1] \leq \varepsilon/2\,.$$

Therefore, by flipping the value of f on all points y such that either $f(y) = 1$ and $y_j = 0$ or $f(y) = 0$ and $y_j = 1$ we obtain a monotone monomial (singleton) g such that $\mathrm{dist}_D(f,g) \leq \varepsilon$, and we have reached a contradiction.

Finally, if

$$\Pr_{y\sim D}[f(y) = 1 \text{ and } y_j = 0] \geq \varepsilon/2$$

[5] We note that the analysis does not explicitly address the case that $T_{f,0} = \emptyset$, where the tester accepts (for every T') simply because $J(T_{f,0})$ is empty. What the analysis implies (implicitly) is that the probability that such an event occurs when $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) > \varepsilon$ is at most a small constant. It is possible to argue this directly (since if $\mathrm{dist}_D(f, \mathcal{MON}_{\mathcal{M}}) > \varepsilon$, then in particular, f is ε-far with respect to D from the all-1 function, so that the probability that no point in $f^{-1}(0)$ is selected in the first sample is very small).

for every $j \in [n]$, then by taking a union bound over all $j \in [n]$ we get the following. With high constant probability over the choice of a sample of size $\Theta(\log n/\varepsilon)$, for each $j \in [n]$, the sample will contain a point y such that $f(y) = 1$ and $y_j = 0$.

4 Distribution-free testing of (general) monomials

The high-level structure of the tester for general monomials is similar to the tester for monotone monomials, but several modifications have to be made (and hence the tester and the notions it is based on are seemingly more complex). In this section we explain what the modifications are and how this affects the analysis.

Recall that for a point $y \in \{0,1\}^n$ we let $O(y) = \{j : y_j = 1\}$ and $Z(y) = \{j : y_j = 0\}$. For a non-empty set of points $Y \subseteq \{0,1\}^n$, let $O(Y) = \bigcap_{y \in Y} O(y)$ and $Z(Y) = \bigcap_{y \in Y} Z(y)$.

4.1 The violation hypergraph (for general monomials)

A basic observation concerning (general) monomials is that if g is a monomial, then for every subset $\{y^0, y^1, \ldots, y^t\}$ such that $g(y^0) = 0$ and $g(y^i) = 1$ for each $1 \le i \le t$, the following holds: There must be at least one index $j \in [n]$ such that either $j \in O(\{y^1, \ldots, y^t\})$ and $y^0_j = 0$, or $j \in Z(\{y^1, \ldots, y^t\})$ and $y^0_j = 1$. This implies that if we have a subset $\{y^0, y^1, \ldots, y^t\}$ such that $g(y^0) = 0$ and $g(y^i) = 1$ for each $1 \le i \le t$, and such that $O(\{y^1, \ldots, y^t\}) \subseteq O(y^0)$ and $Z(\{y^1, \ldots, y^t\}) \subseteq Z(y^0)$, then we have evidence that g is not a monomial. This motivates the next (modified) definition of a violation hypergraph.

Definition 4.1 (Violation hypergraph of f (w.r.t. general monomials))**.** Let $H_f = (V(H_f), E(H_f))$ be the hypergraph whose vertex set, $V(H_f)$ is $\{0,1\}^n$, and whose edge set, $E(H_f)$, contains all subsets $\{y^0, y^1, \ldots, y^t\} \subseteq \{0,1\}^n$, $t \ge 1$, of the following form:

- $f(y^0) = 0$ and $f(y^i) = 1$ for all $1 \le i \le t$.
- $O(\{y^1, ..., y^t\}) \subseteq O(y^0)$. and $Z(\{y^1, ..., y^t\}) \subseteq Z(y^0)$.

Observe that the difference between Definition 3.1 and Definition 4.1 is in the additional requirement that $Z(\{y^1, ..., y^t\}) \subseteq Z(y^0)$ (as well as the fact that $\{1^n\}$ is not an edge in H_f even if $f(1^n) = 0$). Similarly to Lemma 3.2 here we have the next lemma.

Lemma 4.2. *If $E(H_f) = \emptyset$ then f is a monomial.*

Lemma 4.2 follows from the next claim, which is similar to Claim 3.3 (by setting $R = \{0,1\}^n$).

Claim 4.3. *Let R be a subset of $\{0,1\}^n$ and let $H_f(R) = (R, E(H_f(R)))$ be the sub-hypergraph of H_f where $E(H_f(R))$ consists of all edges in H_f that are contained in R. If $E(H_f(R)) = \emptyset$ then there exists $h \in \mathcal{MON}$ such that $h(x) = f(x)$ for every $x \in R$.*

Proof. Based on the values that f assigns to points in R, we next define a monomial h and show that it is consistent with f on R. If $f^{-1}(1) \cap R = \emptyset$, then we let h be the monomial that is the conjunction of x_1 and

$\bar{x}_1$, so that h is the all-0 function. Since $f^{-1}(1) \cap R = \emptyset$, this monomial is consistent with f on all points in R.

Otherwise, let $R_{f,1} = f^{-1}(1) \cap R$ and $R_{f,0} = f^{-1}(0) \cap R$, and let h be the monomial that is the conjunction of all variables x_j such that $j \in O(R_{f,1})$ and all negations of variables $\bar{x}_j$ such that $j \in Z(R_{f,1})$ (where if $O(R_{f,1})$ and $Z(R_{f,1})$ are empty, then h is the all-1 function). We next show that by the premise of the claim, $f(y) = h(y)$ for all $y \in \{0,1\}^n \cap R$.

By the definition of h, for all $y \in R_{f,1}$, $h(y) = 1$. To establish that $h(y) = 0$ for all $y \in R_{f,0}$, assume in contradiction that there is a point $y^0 \in R_{f,0}$ such that $h(y^0) = 1$. Since $h(y^0) = 1$ we have (by the definition of h) that $y^0_j = 1$ for all $j \in O(R_{f,1})$ and $y^0_j = 0$ for all $j \in Z(R_{f,1})$. That is, $Z(R_{f,1}) \subseteq Z(y^0)$ and $O(R_{f,1}) \subseteq O(y^0)$. But this means, by the definition of $H_f(R)$, that $\{y^0 \cup R_{f,1}\}$ is an edge in $H_f(R)$ and we have reached a contradiction. □

The next lemma is analogous to Lemma 3.4. Its proof is the same as the proof of Lemma 3.4 except that the application of Claim 3.3 is replaced by an application of Claim 4.3.

Lemma 4.4. *If $dist_D(f, \mathcal{MON}) > \varepsilon$, then for every vertex cover C of H_f we have $D(C) > \varepsilon$.*

4.2 The tester for general monomials

For a vector $y \in \{0,1\}^n$, and for an index $j \in [n]$, let $y^{\neg j}$ be the same as y except that the j^{th} coordinate in y is flipped. That is, $y^{\neg j}_\ell = y_\ell$ for all $\ell \neq j$ and $y^{\neg j}_j = \bar{y}_j$. For a subset $I \subseteq [n]$ let $y^{\neg I}$ be the vector y with each coordinate $j \in I$ flipped. That is, $y^{\neg I}_\ell = y_\ell$ for all $\ell \notin I$ and $y^{\neg I}_\ell = \bar{y}_\ell$ for all $\ell \in I$. Let $\Delta(y,w) \subseteq [n]$ be the subset of indices j such that $y_j \neq w_j$, and note that $y = w^{\neg\Delta}$ for $\Delta = \Delta(y,w)$.

Algorithm 3. Binary Search for General Monomials (Input: $y \in f^{-1}(0)$, $w \in f^{-1}(1)$)

1. $\Delta \leftarrow \Delta(y,w)$;
2. While $(|\Delta| \geq 2)$ do
   (a) Let (Δ_1, Δ_2) be a fixed partition of Δ where $||\Delta_1| - |\Delta_2|| \leq 1$.
   Specifically, Δ_1 is the set of the first $\lfloor |\Delta|/2 \rfloor$ indices in Δ.
   (b) If $f(w^{\neg\Delta_1}) = 0$, then $\Delta \leftarrow \Delta_1$;
   (c) else if $f(w^{\neg\Delta_2}) = 0$, then $\Delta \leftarrow \Delta_2$;
   (d) else output *fail* and halt.
3. Output the single index $j \in \Delta$;

Figure 3: The binary search procedure for general monomials.

We start by describing the binary search procedure (for general monomials). Its pseudo-code is given in Algorithm 3 (see Figure 3). The procedure receives as input two points $w, y \in \{0,1\}^n$ such that $f(w) = 1$ and $f(y) = 0$ and outputs an index $j \in [n]$ such that $y_j \neq w_j$ and such that $f(w^{\neg j}) = 0$. If f is a monomial, then at least one such index must exist. Note that if $w = 1^n$, then the output of the search is as specified by the binary search procedure for monotone monomials (Algorithm 1). In fact, Algorithm 1 itself (and not only its output specification) is essentially the same as Algorithm 3 for the special case of $w = 1^n$. (Since $f(1^n)$ must equal 1 if f is a monotone monomial, we can think of the binary search procedure for monotone monomials as implicitly working under this assumption.)

The search is performed by repeatedly partitioning a set of indices Δ, starting with $\Delta = \Delta(y,w)$, into two parts Δ_1 and Δ_2 of (almost) equal size, and querying f on the two points, $w^{\neg\Delta_1}$ and $w^{\neg\Delta_2}$. If f returns 1 for both, then the search fails. Otherwise, the search continues with Δ_i for which $f(w^{\neg\Delta_i}) = 0$, unless $|\Delta_i| = 1$, in which case the desired index is found. If the search fails, then we have evidence that f is not a monomial. Namely, we have three points, $w^{\neg\Delta_1}$, $w^{\neg\Delta_2}$ and $w^{\neg\Delta}$, where $\Delta = \Delta_1 \cup \Delta_2$, such that $f(w^{\neg\Delta}) = 0$ and $f(w^{\neg\Delta_1}) = f(w^{\neg\Delta_2}) = 1$. Since $w^{\neg\Delta_1}$ and $w^{\neg\Delta_2}$ disagree on all coordinates in Δ, and all three points agree on all coordinates in $[n] \setminus \Delta$, we have that $Z(\{w^{\neg\Delta_1}, w^{\neg\Delta_2}\}) \subseteq Z(w^{\neg\Delta})$ and $O(\{w^{\neg\Delta_1}, w^{\neg\Delta_2}\}) \subseteq O(w^{\neg\Delta})$, so that the three points constitute an edge in H_f.

Algorithm 4. General Monomials Test

1. Obtain a sample S of $\Theta(1/\varepsilon)$ points, each generated independently according to D.
2. If $S_{f,1} = \emptyset$, then output *accept* and halt. Otherwise, arbitrarily select a point $w \in S_{f,1}$.
3. Obtain a sample T of $\Theta(\sqrt{n}/\varepsilon)$ points, each generated independently according to D.
4. For each point $y \in T_{f,0}$ run the binary search procedure (Algorithm 3) on w, y.
5. If the binary search fails for any of the points, then output *reject* and halt. Otherwise, for each $y \in T_{f,0}$ let $j^w(y)$ be the index returned for y, and let $J^w(T_{f,0}) = \{j^w(y) : y \in T_{f,0}\}$.
6. Obtain another sample T' of size $\Theta(\sqrt{n}/\varepsilon)$ (generated independently according to D).
7. If there is a point $y \in T'_{f,1}$ and an index $j \in J^w(T_{f,0})$ such that $y_j \neq w_j$, then output *reject*, otherwise output *accept*.

Figure 4: The tester for general monomials.

The tester for general monomials starts by obtaining a sample of $\Theta(1/\varepsilon)$ points, each generated independently according to D. The tester arbitrarily selects a point w in this sample that belongs to $f^{-1}(1)$. If no such point exists, then the tester simply accepts f (and halts). Otherwise, this point serves as as a kind of *reference point*. As in the case of the binary search procedure, the tester for monotone monomials (Algorithm 2) is essentially the same as the tester for general monomials (Algorithm 4) with w (implicitly) set to be 1^n.

Next, the tester obtains a sample of $\Theta(\sqrt{n}/\varepsilon)$ points (each generated independently according to D). For each point y in the sample that belongs to $f^{-1}(0)$, the tester performs a binary search on the pair w, y. If any search fails, then the tester rejects (recall that in such a case it has evidence that f is not a monomial). Otherwise, for each point y in the sample that belongs to $f^{-1}(0)$, the tester has an index, $j^w(y) \in \Delta(y,w)$, such that $f(w^{\neg j^w(y)}) = 0$. Let the subset of all these indices be denoted by J. Note that by the construction of J, if f is a monomial, then for every $j \in J$, if $w_j = 1$, then the variable x_j must belong to the conjunction defining f and if $w_j = 0$, then $\bar{x}_j$ must belong to the conjunction.

The tester then takes an additional sample, also of size $\Theta(\sqrt{n}/\varepsilon)$, and checks whether there exists a point y in the sample that belongs to $f^{-1}(1)$ and an index $j \in J$ such that $y_j \neq w_j$. In such a case the tester has evidence that f is not a monomial. Viewing this in terms of the hypergraph H_f, we have that $f(w^{\neg j}) = 0$, $f(y) = f(w) = 1$, and both $Z(\{y,w\}) \subseteq Z(w^{\neg j})$ and $O(\{y,w\}) \subseteq O(w^{\neg j})$, so that $\{w^{\neg j}, y, w\} \in E(H_f)$. The pseudo-code of the tester is given in Figure 4.

4.3 The analysis of the tester for general monomials

We start by stating a very simple claim. This claim explains why the tester can accept f in case it sees no point in the first sample that belongs to $f^{-1}(1)$. It follows from the fact that every monomial that includes a variable and its negation gives a value of 0 to all domain points.

Claim 4.5. *If $D(f^{-1}(1)) \leq \varepsilon$ (that is, f is ε-close to the all-0 function), then $dist_D(f, \mathcal{MON}) \leq \varepsilon$.*

Since the initial sample S consists of $\Theta(1/\varepsilon)$ points (that are generated independently according to D), we get the next corollary.

Corollary 4.6. *If $dist_D(f, \mathcal{MON}) > \varepsilon$, then the probability that Algorithm 4 accepts because $S_{f,1}$ is empty is $\exp(-c)$ where c is the constant in the $\Theta(\cdot)$ notation for S.*

We next modify Definition 3.5.

Definition 4.7 (Empty points and representative indices (for general monomials))**.** For a point $y \in f^{-1}(0)$ and a point $w \in f^{-1}(1)$, we say that y is *empty* with respect to w (and f) if the binary search procedure (Algorithm 3) fails when given w and y as input. We denote the set of empty points with respect to w (and f) by $Y^w_\emptyset(f)$. If y is not empty with respect to w (and f), then we let $j^w(y) \in \Delta(y,w)$ denote the index that the binary search procedure returns. We refer to this index as the *representative* index for y with respect to w. If $y \in Y^w_\emptyset(f)$, then we define $j^w(y)$ to be 0.

Using the notation introduced in Algorithm 4,

$$J^w(T_{f,0}) = \{j^w(y) : y \in T_{f,0} \setminus Y^w_\emptyset(f)\}.$$

For a subset of indices $J \subseteq [n]$, let $Y^w_{f,1}(J)$ denote the set of points $y \in f^{-1}(1)$ such that $y_j \neq w_j$ for some $j \in J$, and let $Y^w_{f,0}(J)$ denote the set of points $y \in f^{-1}(0)$ for which $j^w(y) \in J$. Lemma 3.6 and Claim 3.7 are adapted as follows.

Lemma 4.8. *Suppose $dist_D(f, \mathcal{MON}) > \varepsilon$ and consider any fixed point $w \in f^{-1}(1)$ and a sample T of $c_1\sqrt{n}/\varepsilon$ points generated independently according to D. For a sufficiently large constant c_1, with probability at least $5/6$ over the choice of T, either $T_{f,0}$ contains an empty point with respect to w (and f) or $D(Y^w_{f,1}(J(T_{f,0}))) \geq \varepsilon/(4\sqrt{n})$.*

Claim 4.9. *Let I be any fixed subset of $[n]$, let w be any fixed point in $f^{-1}(1)$, and consider a sample T of $s = c_1\sqrt{n}/\varepsilon$ points generated independently according to D. For a sufficiently large constant c_1, with probability at least $9/10$ over the choice of T, either*

$$|J^w(T_{f,0}) \setminus I| \geq \sqrt{n} \quad or \quad D(Y^w_{f,0}([n] \setminus (J^w(T_{f,0}) \cup I))) \leq \varepsilon/2.$$

The proof of Claim 4.9 is exactly the same as the proof of Claim 3.7 except that each instance of $J(T_{f,0})$ needs to be replaced by $J^w(T_{f,0})$, and each instance of the form $Y_{f,0}(\cdot)$ needs to be replaced by $Y^w_{f,0}(\cdot)$. The proof of Lemma 4.8 is also essentially the same as the proof of Lemma 3.6. However, there is one subtle difference (due to the difference in the definitions of the violation hypergraphs for monotone and general monomials) and hence we only give the details of the difference.

Proof of Lemma 4.8. The subset C is constructed in the same manner as in the proof of Lemma 3.6 except that $Y_\emptyset(f)$ is replaced by $Y^w_\emptyset(f)$, each instance of $Y_{f,1}(\cdot)$ is replaced by $Y^w_{f,1}(\cdot)$, the application of Claim 3.7 is replaced by an application of Claim 4.9 (and $J(T_{f,0})$ and $Y_{f,0}(\cdot)$ are replaced by $J^w(T_{f,0})$ and $Y^w_{f,0}(\cdot)$, respectively, as in the proof of Claim 4.9). The argument bounding the weight of C is also the same as in the proof of Lemma 3.6, and so it only remains to give the details establishing that C is indeed a vertex cover of H_f (as defined in Definition 4.1).

Consider any point $y \in f^{-1}(0)$ that is contained in at least one edge in H_f. If $y \in Y^w_\emptyset(f)$, then it is added to the cover in the first stage. Otherwise, y has a representative index $j^w(y) \in \Delta(y,w)$. Consider the iterations in the second stage of the construction of C. If for some iteration ℓ we have that $j^w(y) \in J^w(T^\ell_{f,0})$, then the cover contains all points in $Y^w_{f,1}(\{j^w(y)\}) \subseteq Y^w_{f,1}(J^w(T^\ell_{f,0}))$. By the definition of H_f, each edge in H_f that contains y must contain some $y' \in f^{-1}(1)$ such that $y'_{j^w(y)} = y_{j^w(y)}$ (otherwise, for some edge B, it would not be true that $Z(B \setminus \{y\}) \subseteq Z(y)$ and $O(B \setminus \{y\}) \subseteq O(y)$). But for each such y', since $y_{j^w(y)} \neq w_{j^w(y)}$, we have that $y'_{j^w(y)} \neq w_{j^w(y)}$, and so $y' \in Y^w_{f,1}(\{j^w(y)\})$. Therefore, all edges containing y are covered after iteration ℓ. On the other hand, if $j^w(y) \notin J^w(T^\ell_{f,0})$ for every iteration ℓ, then y is added to C in the third stage (where the argument is as in the proof of Lemma 3.6). $\square$

Lemma 4.8 together with Corollary 4.6 imply the next theorem.

Theorem 4.10. *Algorithm 4 is a distribution-free 1-sided-error tester for (membership in) $\mathcal{MON}$. Its query complexity is $O(\sqrt{n}\log n/\varepsilon)$.*

The proof of Theorem 4.10 is essentially the same as the proof of Theorem 3.8, where the application of Lemma 3.6 is replaces by an application of Lemma 4.8, and the probability that the tester erroneously accepts due to $S_{1,f} = \emptyset$ is taken into account (using Corollary 4.6).

4.4 Disallowing the all-1 function

Let $\mathcal{MON}'$ denote the class of monomials that are a conjunction of at least one literal, so that $\mathcal{MON}' = \mathcal{MON} \setminus \{\mathbf{1}\}$ (where $\mathbf{1}$ is the all-1 function). It is possible to augment our tester for $\mathcal{MON}$ (Algorithm 4) so that it work for $\mathcal{MON}'$ in a very similar manner to what was shown for monotone monomials in Subsection 3.4. The only difference is that when we take the additional sample R, we check whether there exists at least one index $j \in [n]$ such that for some $b \in \{0,1\}$, $y_j = b$ for all points $y \in S \cap f^{-1}(1)$. The modification in the analysis is that we show that if $\mathrm{dist}_D(f,\mathbf{1}) \le \varepsilon/2$ and $\mathrm{dist}_D(f,\mathcal{MON}') > \varepsilon$, then

$$\Pr_{y\sim D}[f(y) = 1 \text{ and } y_j = b] \ge \varepsilon/2$$

for every $j \in [n]$ and $b \in \{0,1\}$.

Acknowledgements

We would like to thank several anonymous reviewers of this paper for their helpful comments. In particular, we would like to thank a reviewer of the extended abstract version of this paper for suggesting to simplify the proof of Lemma 3.4.

References

[1] MANUEL BLUM, MICHAEL LUBY, AND RONITT RUBINFELD: Self-testing/correcting with applications to numerical problems. *J. Comput. System Sci.*, 47(3):549–595, 1993. [doi:10.1016/0022-0000(93)90044-W] 156

[2] H. CHERNOFF: A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23(4):493–507, 1952. [doi:10.1214/aoms/1177729330] 166

[3] ILIAS DIAKONIKOLAS, HOMIN K. LEE, KEVIN MATULEF, KRZYSZTOF ONAK, RONITT RUBINFELD, ROCCO A. SERVEDIO, AND ANDREW WAN: Testing for concise representations. In *Proc. 48th FOCS*, pp. 549–557. IEEE Comp. Soc. Press, 2007. [doi:10.1109/FOCS.2007.32] 156, 158

[4] YEVGENIY DODIS, ODED GOLDREICH, ERIC LEHMAN, SOFYA RASKHODNIKOVA, DANA RON, AND ALEX SAMORODNITSKY: Improved testing algorithms for monotonocity. In *Proc. 3rd Intern. Workshop Random. and Approx. Tech. Comput. Sci. (RANDOM)*, number 1671 in Lecture Notes in Comput. Sci., pp. 97–108. Springer, 1999. [doi:10.1007/978-3-540-48413-4_10] 156, 158

[5] FUNDA ERGÜN, SAMPATH KANNAN, S. RAVI KUMAR, RONITT RUBINFELD, AND MAHESH VISWANATHAN: Spot-checkers. *J. Comput. System Sci.*, 60(3):717–751, 2000. [doi:10.1006/jcss.1999.1692] 156, 158

[6] ELDAR FISCHER, GUY KINDLER, DANA RON, SHMUEL SAFRA, AND ALEX SAMORODNITSKY: Testing juntas. *J. Comput. System Sci.*, 68(4):753–787, 2004. [doi:10.1016/j.jcss.2003.11.004] 156, 158

[7] DANA GLASNER AND ROCCO A. SERVEDIO: Distribution-free testing lower bounds for basic Boolean functions. *Theory of Computing*, 5(10):191–216, 2009. [doi:10.4086/toc.2009.v005a010] 156, 157, 159

[8] ODED GOLDREICH, SHAFI GOLDWASSER, ERIC LEHMAN, DANA RON, AND ALEX SAMORODINSKY: Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000. [doi:10.1007/s004930070011] 158

[9] ODED GOLDREICH, SHAFI GOLDWASSER, AND DANA RON: Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. [doi:10.1145/285055.285060] 156

[10] SHIRLEY HALEVY AND EYAL KUSHILEVITZ: Distribution-free property-testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007. [doi:10.1137/050645804] 156, 157, 158

[11] SHIRLEY HALEVY AND EYAL KUSHILEVITZ: Distribution-free connectivity testing for sparse graphs. *Algorithmica*, 51(1):24–48, 2008. [doi:10.1007/s00453-007-9054-1] 156, 158

[12] SHIRLEY HALEVY AND EYAL KUSHILEVITZ: Testing monotonicity over graph products. *Random Structures Algorithms*, 33(1):44–67, 2008. [doi:10.1002/rsa.20211] 156, 158

[13] SWASTIK KOPPARTY AND SHUBHANGI SARAF: Tolerant linearity testing and locally testable codes. In *Proc. 13th Intern. Workshop Random. and Approx. Tech. Comput. Sci. (RANDOM)*, number 5687 in Lecture Notes in Comput. Sci., pp. 601–614. Springer, 2009. [doi:10.1007/978-3-642-03685-9_45] 158

[14] KEVIN MATULEF, RYAN O'DONNELL, RONITT RUBINFELD, AND ROCCO A. SERVEDIO: Testing halfspaces. *SIAM J. Comput.*, 39(5):2004–2047, 2010. [doi:10.1137/070707890] 156

[15] MICHAL PARNAS, DANA RON, AND RONITT RUBINFELD: Tolerant property testing and distance approximation. *J. Comput. System Sci.*, 72(6):1012–1042, 2006. [doi:10.1016/j.jcss.2006.03.002] 158

[16] MICHAL PARNAS, DANA RON, AND ALEX SAMORODNITSKY: Testing basic Boolean formulae. *SIAM J. Discrete Math.*, 16(1):20–46, 2002. [doi:10.1137/S0895480101407444] 156, 157

[17] RONITT RUBINFELD AND MADHU SUDAN: Robust characterization of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. [doi:10.1137/S0097539793255151] 156

[18] GYÖRGY TURÁN: Lower bounds for PAC learning with queries. In *Proc. 6th Ann. ACM Conf. Comput. Learn. Theory (COLT)*, pp. 384–391. ACM Press, 1993. [doi:10.1145/168304.168382] 157

[19] LESLIE G. VALIANT: A theory of the learnable. *CACM*, 27(11):1134–1142, November 1984. [doi:10.1145/1968.1972] 156

AUTHORS

Elya Dolev
Tel Aviv University, Tel Aviv, Israel
elyadolev@msn.com

Dana Ron
Professor
Tel Aviv University, Tel Aviv, Israel
danar@eng.tau.ac.il
http://www.eng.tau.ac.il/~danar

ABOUT THE AUTHORS

ELYA DOLEV received her M. Sc. from Tel Aviv University in 2011. She wrote her M. Sc. thesis under the supervision of Dana Ron. Her research interests include sublinear approximation algorithms and in particular property testing.

DANA RON received her Ph. D. from the Hebrew University, Jerusalem, in 1995. Between 1995 and 1997 she was an NSF Postdoc at MIT. During the academic year 1997-98 she was a science scholar at the Mary Ingraham Bunting Institute (Radcliffe College) and at MIT. Since 1998 she has been a faculty member at the Faculty of Engineering, Tel Aviv University. During the academic year 2003-04 she was a fellow at the Radcliffe Institute for Advanced Study, Harvard University, Her research focuses on sublinear approximation algorithms and in particular on property testing.