Distribution-Free Testing Lower Bounds for Basic Boolean Functions

Dana Glasner[1]* and Rocco A. Servedio[1]**

Department of Computer Science
Columbia University
New York, NY 10027, USA
{dglasner,rocco}@cs.columbia.edu

Abstract. In the *distribution-free* property testing model, the distance between functions is measured with respect to an arbitrary and unknown probability distribution $\mathcal{D}$ over the input domain. We consider distribution-free testing of several basic Boolean function classes over $\{0,1\}^n$, namely monotone conjunctions, general conjunctions, decision lists, and linear threshold functions. We prove that for each of these function classes, $\Omega((n/\log n)^{1/5})$ oracle calls are required for any distribution-free testing algorithm. Since each of these function classes is known to be distribution-free properly learnable (and hence testable) using $\Theta(n)$ oracle calls, our lower bounds are within a polynomial factor of the best possible.

1 Introduction

The field of property testing deals with algorithms that decide whether an input object has a certain property or is far from having the property after reading only a small fraction of the object. Property testing was introduced in [21] and has evolved into a rich field of study (see [3, 7, 10, 19, 20] for some surveys). A standard approach in property testing is to view the input to the testing algorithm as a function over some finite domain; the testing algorithm is required to distinguish functions that have a certain property P from functions that are ϵ-far from having property P. In the most commonly considered property testing scenario, a function f is ϵ-far from having a property P if f disagrees with every function g that has property P on at least an ϵ fraction of the points in the input domain; equivalently, the distance between functions f and g is measured with respect to the uniform distribution over the domain. The testing algorithm "reads" f by adaptively querying a black-box oracle for f at points x of the algorithm's choosing (such oracle calls are often referred to as "membership queries" in computational learning theory). The main goal in designing property testing algorithms is to use as few queries as possible to distinguish the two types of functions; ideally the number of queries should depend only on ϵ and should be independent of the size of f's domain.

One can of course view any property P as a class of functions (the class of those functions that have property P). In recent years there has been considerable work in

* Supported in part by an FFSEAS Presidential Fellowship.
** Supported in part by NSF award CCF-0347282, by NSF award CCF-0523664, and by a Sloan Foundation Fellowship.

the standard "uniform distribution" property testing scenario on testing various natural properties of Boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$, i.e. testing various Boolean function classes. Some classes for which uniform distribution testing results have been obtained are monotone functions [6, 9, 12]; Boolean literals, monotone conjunctions, general conjunctions and s-term monotone DNFs [18]; J-juntas [8]; parity functions (which are equivalent to degree-1 polynomials) [4]; degree-d polynomials [2]; decision lists, s-term DNFs, size-s decision trees and s-sparse polynomials [5]; and linear threshold functions [17].

Distribution-free property testing. A natural generalization of property testing is to consider a broader notion of the distance between functions. Given a probability distribution $\mathcal{D}$ over the domain, we may define the distance between f and g as the probability that an input x drawn from $\mathcal{D}$ has $f(x) \neq g(x)$; the "standard" notion of property testing described above corresponds to the case where $\mathcal{D}$ is the uniform distribution. *Distribution-free property testing* is the study of property testers in a setting where distance is measured with respect to a *fixed but unknown and arbitrary* probability distribution $\mathcal{D}$. Since the distribution $\mathcal{D}$ is unknown, in this scenario the testing algorithm is allowed to draw random samples from $\mathcal{D}$ in addition to querying a black-box oracle for the value of the function.

Distribution-free property testing is well-motivated by very similar models in computational learning theory (namely the model of distribution-free PAC learning with membership queries, which is closely related to the well-studied model of exact learning from equivalence and membership queries), and by the fact that in various settings the uniform distribution may not be the best way to measure distances. Distribution-free property testing has been considered by several authors [1, 11, 13–15]; we briefly describe some of the most relevant prior work below.

Goldreich *et al.* [11] introduced the model of distribution-free property testing, and observed that any *proper* distribution-free PAC learning algorithm (such a learning algorithm for a class of functions always outputs a hypothesis function that itself belongs to the class) can be used as a distribution-free property testing algorithm. They also showed that several graph properties that have testing algorithms with query complexity independent of input size in the uniform-distribution model (such as bipartiteness, k-colorability, ρ-clique, ρ-cut and ρ-bisection) do not have distribution-free testing algorithms with query complexity independent of input size. In contrast, Halevy and Kushilevitz [14] gave a distribution-free algorithm for testing connectivity in sparse graphs that has poly$(1/\epsilon)$ query complexity independent of input size.

A range of positive and negative results have been established for distribution-free testing of Boolean functions over $\{0,1\}^n$. [15] showed that any distribution-free monotonicity testing algorithm over $\{0,1\}^n$ must make $2^{\Omega(n)}$ queries; this is in contrast with the uniform distribution setting, where monotonicity testing algorithms are known that have query complexity poly$(n, 1/\epsilon)$ [6, 9, 12]. On the other hand, [13] showed that for several important function classes over $\{0,1\}^n$ such as juntas, parities, low-degree polynomials and Boolean literals, there exist distribution-free testing algorithms with query complexity poly$(1/\epsilon)$ independent of n; these distribution-free results match the query bounds of uniform distribution testing algorithms for these classes.

To sum up, the current landscape of distribution-free property testing is intriguingly varied. For some testing problems (juntas, parities, Boolean literals, low-degree polynomials, connectivity in sparse graphs) the complexity of distribution-free testing is known to be essentially the same as the complexity of uniform-distribution testing; but for other natural testing problems (monotonicity, bipartiteness, k-colorability, ρ-clique, ρ-cut, ρ-bisection), distribution-free testing provably requires many more queries than uniform-distribution testing.

This work. Our work is motivated by the fact that for many Boolean function classes over $\{0,1\}^n$ that are of fundamental interest, a very large gap exists between the query complexities of the best known distribution-free property testing algorithms (which typically follow trivially from learning algorithms and have query complexity $\Omega(n)$) and the best known uniform distribution property testing algorithms (which typically have query complexity poly$(1/\epsilon)$ independent of n). A natural goal is to try to close this gap, either by developing efficient distribution-free testing algorithms or by proving lower bounds for distribution-free testing for these classes.

We study distribution-free testability of several fundamental classes of Boolean functions that have been previously considered in the uniform distribution testing framework, and have been extensively studied in various distribution-free learning models. More precisely, we consider the following classes (in order of increasing generality): monotone conjunctions, arbitrary conjunctions, decision lists, and linear threshold functions. Each of these four classes is known to be testable in the uniform distribution setting using poly$(1/\epsilon)$ many queries, independent of n (see [18] for monotone and general conjunctions, [5] for decision lists, and [17] for linear threshold functions). On the other hand, for each of these classes the most efficient known distribution-free testing algorithm is simply to use a proper learning algorithm. Using the fact that each of these classes has Vapnik-Chervonenkis dimension $\Theta(n)$, standard results in learning theory yield well-known algorithms that use $O(n/\epsilon)$ random examples and no membership queries (see e.g. Chapter 3 of [16]), and known results also imply that any learning algorithm must make $\Omega(n)$ oracle calls (see [22]).

Our main results are strong distribution-free lower bounds for testing each of these four function classes:

Theorem 1. *Let T be any algorithm which, given oracle access to an unknown $f : \{0,1\}^n \to \{0,1\}$ and (sampling) oracle access to an unknown distribution $\mathcal{D}$ over $\{0,1\}^n$, tests whether f is a monotone conjunction versus $\Theta(1)$-far from every monotone conjunction with respect to $\mathcal{D}$. Then T must make $\Omega((n/\log n)^{1/5})$ oracle calls in total. The same lower bound holds for testing general conjunctions, testing decision lists, and testing linear threshold functions.*

These results show that for these function classes, distribution-free testing is nearly as difficult (from a query perspective) as distribution-free learning, and is much more difficult than uniform-distribution testing.

Organization. After giving preliminaries in Section 2, in Section 3 we present our construction of "yes" and "no" (function, distribution) pairs that are used in the lower bound for monotone conjunctions. The actual lower bound proof is given in Section 4. In Section 5 we give a simple argument that extends the result to a lower bound for arbitrary conjunctions and for decision lists. In Appendix B we describe a variant of the construction for linear threshold functions, and in Appendix C we use it to prove the lower bound for linear threshold functions.

2 Preliminaries

Throughout the paper we deal with Boolean functions over n input variables.

Definition 1. *Let $\mathcal{D}$ be a probability distribution over $\{0,1\}^n$. Given Boolean functions $f, g : \{0,1\}^n \to \{0,1\}$, the* distance between f and g with respect to $\mathcal{D}$ *is defined by $dist_{\mathcal{D}}(f,g) \overset{def}{=} \mathbf{Pr}_{x\sim\mathcal{D}}[f(x) \neq g(x)]$.*

If C is a class of Boolean functions over $\{0,1\}^n$, we define the distance between f and C with respect to $\mathcal{D}$ *to be $dist_{\mathcal{D}}(f,C) \overset{def}{=} \min_{g\in C} dist_{\mathcal{D}}(f,g)$.*

We say that f is ϵ-far from C with respect to $\mathcal{D}$ *if $dist_{\mathcal{D}}(f,C) \geq \epsilon$.*

Now we can define the notion of a distribution-free tester for a class of functions C:

Definition 2. *A* distribution-free tester for class C *is a probabilistic oracle machine T which takes as input a distance parameter $\epsilon > 0$, is given access to*

– *a* black-box oracle *to a fixed (but unknown and arbitrary) function $h : \{0,1\}^n \to \{0,1\}$ (when invoked with input x, the oracle returns the value $h(x)$); and*
– *a* sampling oracle *for a fixed (but unknown and arbitrary) distribution $\mathcal{D}$ over $\{0,1\}^n$ (each time it is invoked this oracle returns a pair $(x, h(x))$ where x is independently drawn from $\mathcal{D}$),*

and satisfies the following two conditions: for any $h : \{0,1\}^n \to \{0,1\}$ and any distribution $\mathcal{D}$,

– *If h belongs to C, then $\mathbf{Pr}[T^{h,\mathcal{D}} = Accept] \geq \frac{2}{3}$; and*
– *If h is ϵ-far from C w.r.t. $\mathcal{D}$, then $\mathbf{Pr}[T^{h,\mathcal{D}} = Accept] \leq \frac{1}{3}$.*

This definition allows the tester to be adaptive and to have two-sided error; this is of course the strongest version for proving lower bounds.

The classes we consider. For completeness we define here all the classes of functions that we will consider: these are (in order of increasing generality) monotone conjunctions, general conjunctions, decision lists, and linear threshold functions. We note that each of these function classes is quite basic and natural and has been studied intensively in fields such as computational learning theory.

The class $MCONJ$ consists of all monotone conjunctions of any set of Boolean variables from $x_1, \ldots, x_n$, i.e. all ANDs of (unnegated) Boolean variables.

The class $CONJ$ consists of all conjunctions of any set of Boolean literals over $\{0,1\}^n$ (a literal is a Boolean variable or the negation of a variable).

A *decision list* L of length k over the Boolean variables $x_1, \ldots, x_n$ is defined by a list of k pairs and a bit $(\ell_1, \beta_1), (\ell_2, \beta_2), \ldots, (\ell_k, \beta_k), \beta_{k+1}$ where each ℓ_i is a Boolean literal and each β_i is either 0 or 1. Given any $x \in \{0,1\}^n$, the value of $L(x)$ is β_i if i is the smallest index such that ℓ_i is made true by x; if no ℓ_i is true then $L(x) = \beta_{k+1}$. Let DL denote the class of all decision lists of arbitrary length $k \geq 0$ over $\{0,1\}^n$.

A *linear threshold function* is defined by a list of $n+1$ real values $w_1, \dots, w_n, \theta$. The value of the function on input $x \in \{0,1\}^n$ is 1 if $w_1 x_1 + \cdots + w_n x_n \geq \theta$ and is 0 if $w_1 x_1 + \cdots + w_n x_n < \theta$. We write LTF to denote the class of all linear threshold functions over $\{0,1\}^n$.

It is well known and easy to see that $MCONJ \subsetneq CONJ \subsetneq DL \subsetneq LTF$.

Notation. For a string $x \in \{0,1\}^n$ we write x_i to denote the i-th bit of x. For $x, y \in \{0,1\}^n$ we write $x \wedge y$ to denote the n-bit string z which is the bitwise AND of x and y, i.e. $z_i = x_i \wedge y_i$ for all i. The string $x \vee y$ is defined similarly to be the bitwise OR of x and y.

Recall that the *total variation distance*, or *statistical distance*, between two random variables X and Y that take values in a finite set S is $d_{TV}(X,Y) \overset{\text{def}}{=} \frac{1}{2}\sum_{\zeta \in S} |\mathbf{Pr}[X = \zeta] - \mathbf{Pr}[Y = \zeta]|$.

3 The two distributions for monotone conjunctions

In this section we define two distributions, $\mathcal{YES}$ and $\mathcal{NO}$, over pairs $(h, \mathcal{D})$ where $h : \{0,1\}^n \to \{0,1\}$ is a Boolean function and $\mathcal{D}$ is a distribution over the domain $\{0,1\}^n$. We will prove that these distributions have the following properties:

1. For every pair $(g, \mathcal{D}_g)$ in the support of $\mathcal{YES}$, the function g is a monotone conjunction (and hence any tester for $MCONJ$ must accept every such pair with probability at least $2/3$).
2. For every pair $(f, \mathcal{D}_f)$ in the support of $\mathcal{NO}$, the function f is $1/3$-far from $MCONJ$ with respect to $\mathcal{D}_f$ (and hence any tester for $MCONJ$ must accept every such pair with probability at most $1/3$).

Our constructions are parameterized by three values ℓ, m and s. As we will see the optimal setting of these parameters (up to multiplicative constants) for our purposes is

$$\ell \overset{\text{def}}{=} n^{2/5}(\log n)^{3/5}, \quad m \overset{\text{def}}{=} (n/\log n)^{2/5}, \quad s \overset{\text{def}}{=} \log n. \tag{1}$$

To keep the different roles of these parameters clear in our exposition we will present our constructions and analyses in terms of "ℓ," "m" and "s" as much as possible and only plug in the values from (1) toward the end of our analysis.

3.1 The $\mathcal{YES}$ distribution.

A draw from the distribution $\mathcal{YES}$ over $(g, \mathcal{D}_g)$ pairs is obtained as follows:

– Let $R \subset [n]$ be a set of size $2\ell m$ selected uniformly at random. Randomly partition the set R into $2m$ subsets $A_1, B_1, \dots, A_m, B_m$, each of size ℓ. Let $a^i \in \{0,1\}^n$ be the string whose j-th bit is 0 iff $j \in A_i$. The string b^i is defined similarly. The string c^i is defined to be $a^i \wedge b^i$, and similarly we define the set $C_i = A_i \cup B_i$. We sometimes refer to a^i, b^i, c^i as the "points of the i-th block."

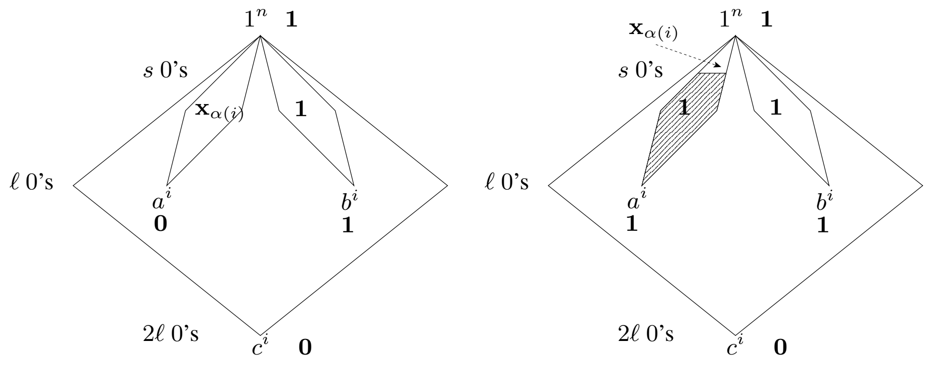

Fig. 1. The left figure shows how a yes-function g labels c^i and the points above it (including a^i and b^i). Bold print indicate the label that g assigns. Note that every point above b^i is labeled $\mathbf{1}$ by g, and points above a^i are labeled according to $\mathbf{x}_{\alpha(i)}$. The right figure shows how a no-function f labels c^i and the points above it (including a^i and b^i). Again, bold print indicates the label that f assigns. Note that every point above b^i is labeled $\mathbf{1}$ by f, and points above a^i with less than s 0's are labeled according to $\mathbf{x}_{\alpha(i)}$. The i-special points for block i are shaded and are labeled $\mathbf{1}$ by f.

– Let g_1 be the conjunction of all variables in $[n] \setminus R$.
– For each $i = 1, \ldots, m$ let $\alpha(i)$ be an element chosen uniformly at random from the set A_i; we say that $\alpha(i)$ is a *representative* of A_i. Let g_2 be a conjunction of length m formed by taking $g = x_{\alpha(1)} \wedge \cdots \wedge x_{\alpha(m)}$, i.e. g is an AND of the representatives from each of $A_1, \ldots, A_m$.
– The function g is taken to be $g = g_1 \wedge g_2$. For each $i = 1, \ldots, m$ the distribution $\mathcal{D}_g$ puts weight $2/(3m)$ on b^i and puts weight $1/(3m)$ on c^i.

It is clear that for every $(g, \mathcal{D}_g)$ in the support of $\mathcal{YES}$, the function g is a monotone conjunction that contains exactly $n - 2m\ell + m$ variables, so Property (1) indeed holds.

3.2 The $\mathcal{NO}$ distribution.

A draw from the distribution $\mathcal{NO}$ of $(f, \mathcal{D}_f)$ pairs is obtained as follows:

– As in the yes-case, let $R \subset [n]$ be a randomly selected set of size $2\ell m$, and randomly partition the set R into $2m$ subsets $A_1, B_1, \ldots, A_m, B_m$, each of size ℓ. The points a^i, b^i, c^i and sets A_i, B_i, C_i are defined as in the yes-case. The distribution $\mathcal{D}_f$ is uniform over the $3m$ points $a^1, \ldots, c^m$.
– Construct the conjunctions g_1 and g_2 exactly as in the yes-case: g_1 is the conjunction of all variables in $[n] \setminus R$ and g_2 is $x_{\alpha(1)} \wedge \cdots \wedge x_{\alpha(m)}$ where each $\alpha(i)$ is a representative chosen uniformly from A_i.
– Define the function f' as follows: $f'(x) = 0$ if there exists some $i \in [m]$ such that both the following conditions hold:
  • $x_{\alpha(i)} = 0$ and

- (fewer than s of the elements $j \in A_i$ have $x_j = 0$) or ($x_j = 0$ for some $j \in B_i$).

  The following terminology will be useful: we say that an input $x \in \{0,1\}^n$ is i-*special* if (at least s elements $j \in A_i$ have $x_j = 0$) and ($x_j = 1$ for all $j \in B_i$). Thus an equivalent way to define f' is that $f'(x) = g_2(x)$ unless $g_2(x) = 0$ (because some $x_{\alpha(i)} = 0$) but x is i-special for each i such that $x_{\alpha(i)} = 0$; in this case $f'(x) = 1$.

– The final function f is defined as $f = g_1 \wedge f'$.

It is easy to see that in both the yes-case and the no-case, any black-box query that sets any variable in $[n] \setminus R$ to 0 will give a 0 response. To give some intuition for our construction, let us explain here the role that the large conjunction g_1 (over $n - 2\ell m$ variables) plays in both the $\mathcal{YES}$ and $\mathcal{NO}$ constructions. The idea is that because of g_1, a testing algorithm that has obtained strings $z^1, \ldots, z^q$ from the distribution $\mathcal{D}$ will "gain nothing" by querying any string x that has any bit x_i set to 0 that was set to 1 in all of $z^1, \ldots, z^q$. This is because such a variable x_i will with very high probability (over a random choice of $(f, \mathcal{D}_f)$ from $\mathcal{NO}$ or a random choice of $(g, \mathcal{D}_g)$ from $\mathcal{YES}$) be contained in g_1, so in both the "yes" and "no" cases the query will yield an answer of 0 with very high probability. Consequently there is no point in making such a query in the first place. (We give a rigorous version of this argument in Section 4.2.)

For any $(f, \mathcal{D}_f)$ drawn from $\mathcal{NO}$, we have $f(a^i) = f(b^i) = 1$ and $f(c^i) = 0$ for each $i = 1, \ldots, m$. It is noted in [18] (and is easy to check) that any monotone conjunction h must satisfy $h(x) \wedge h(y) = h(x \wedge y)$ for all $x, y \in \{0,1\}^n$, and thus must satisfy $h(c^i) = h(a^i) \wedge h(b^i)$. Thus any monotone conjunction h must disagree with f on at least one of a_i, b_i, c_i for all i, and consequently f is $1/3$-far from any monotone conjunction with respect to $\mathcal{D}_f$.

Thus we have established properties (1) and (2) stated at the beginning of this section. These give:

Lemma 1. *Any distribution-free tester for $MCONJ$ that is run with distance parameter $\epsilon = 1/3$ must accept a random pair $(g, \mathcal{D}_g)$ drawn from $\mathcal{YES}$ with probability at least $2/3$, and must accept a random pair $(f, \mathcal{D}_f)$ drawn from $\mathcal{NO}$ with probability at most $1/3$.*

4 The lower bound for monotone conjunctions

In this section we will prove the following theorem:

Theorem 2. *Let $q \overset{def}{=} \frac{1}{20}(\frac{n}{\log n})^{1/5}$. Let T be any probabilistic oracle algorithm that, given a pair $(h, \mathcal{D})$, makes at most q black-box queries to h and samples $\mathcal{D}$ at most q times. Then we have*

$$\left|\mathbf{Pr}_{(g,\mathcal{D}_g)\sim\mathcal{YES}}[T^{g,\mathcal{D}_g} = \textit{Accept}] - \mathbf{Pr}_{(f,\mathcal{D}_f)\sim\mathcal{NO}}[T^{f,\mathcal{D}_f} = \textit{Accept}]\right| \le \frac{1}{4}.$$

Note that in the above theorem each probability is taken over the draw of the (function,distribution) pair from the appropriate distribution $\mathcal{YES}$ or $\mathcal{NO}$, over the random draws from the distribution $\mathcal{D}_f$ or $\mathcal{D}_g$, and over any internal randomness of algorithm T. Lemma 1 and Theorem 2 together immediately imply the first part of Theorem 1, our lower bound for monotone conjunctions.

4.1 The idea.

Here is some high-level intuition for the proof. If T could find a^i, b^i and c^i for some i then T would know which case it is in (yes versus no), because $h(a^i) \wedge h(b^i) = h(c^i)$ if and only if T is in the yes-case. Since T can only make $q \ll \sqrt{m}$ draws from $\mathcal{D}$, the birthday paradox tells us that with high probability the random sample that T draws contains at most one of a^i, b^i and c^i for each i. The c^i-type points (with $n - 2\ell$ ones) are labeled negative in both the yes- and no- cases, so these "look the same" to T in both cases. And since the distributions $\mathcal{D}_g$ (in the yes-case) and $\mathcal{D}_f$ (in the no-case) put weight only on the positive a^i and b^i-type points (with $n - \ell$ ones), these points "look the same" to T as well in both cases. So with high probability T cannot distinguish between yes-pairs and no-pairs on the basis of the first q random draws alone. (Corollary 1 formalizes this intuition.)

Of course, though, T can also make q queries. Can T perhaps identify a triple (a^i, b^i, c^i) through these queries, or perhaps T can otherwise determine which case it is in even without finding a triple? The crux of the proof is to show that in fact queries actually cannot help T much; we now sketch the idea.

Consider a single fixed block $i \in [m]$. If none of a^i, b^i or c^i are drawn in the initial sample, then by the argument of Section 3.2 the tester will get no useful information about which case (s)he is in from this block. By the birthday paradox we can assume that at most one of a^i, b^i and c^i is drawn in the initial sample; we consider the three cases in turn.

If b^i is drawn, then by the Section 3.2 argument all query points will have all the bits in A_i set to 1; such queries will "look the same" in both the yes- and no- cases as far as the i-th block is concerned.

If a^i is drawn (so we are in the no-case), then by the Section 3.2 argument all query points will have all the bits in B_i set to 1. Using the definition of f', as far as the i-th block is concerned with high probability it will "look like" the initial a^i point was a b^i-point from the yes-case. This is because the only way the tester can tell that it is in the no-case is if it manages to query a point which has fewer than s bits from A_i set to 0 but the representative $\alpha(i)$ is one of those bits. Such points are hard to find since $\alpha(i)$ is randomly selected from A_i. (See the "a-witness" case in the proof of Lemma 6.)

Finally, suppose that c^i is drawn. The only way a tester can distinguish between the yes- and no- cases is by finding an i-special point (or determining that no such point exists), but to find such a point it must make a query with at least s 0's in C_i, all of which lie in A_i. This is hard to do since the tester does not know how the elements of C_i are divided into the sets A_i and B_i. (See the "c-witness" case in the proof of Lemma 6.)

4.2 Proof of Theorem 2.

Fix any probabilistic oracle algorithm T that makes at most q black-box queries to h and samples $\mathcal{D}$ at most q times. Without loss of generality we may assume that T first makes exactly q draws from distribution $\mathcal{D}$, and then makes exactly q (adaptive) queries to the black-box oracle for h.

It will be convenient for us to assume that algorithm T is actually given "extra information" on certain draws from the distribution $\mathcal{D}$. More precisely, we suppose that each time T calls the oracle for $\mathcal{D}$,

– If a "c^i-type" labeled example $(c^i, h(c^i))$ is generated by the oracle, algorithm T receives the triple $(c^i, h(c^i), \alpha(i))$ (recall that $\alpha(i)$ is the index of the variable from C_i that belongs to the conjunction g_2);
– If a "non-c^i-type" labeled example $(x, h(x))$ is generated by the oracle where $x \neq c^i$ for all $i = 1, \ldots, m$, algorithm T receives the triple $(x, h(x), 0)$. (Thus there is no "extra information" given on non-c^i points.)

It is clear that proving Theorem 2 for an arbitrary algorithm T that receives this extra information establishes the original theorem as stated (for algorithms that do not receive the extra information).

Following [15], we now define a *knowledge sequence* to precisely capture the notion of "what an algorithm learns from its queries." A knowledge sequence is a sequence of elements corresponding to the interactions that an algorithm has with each of the two oracles. The first q elements of a knowledge sequence are triples as described above; each corresponds to an independent draw from the distribution $\mathcal{D}$. The remaining elements of the knowledge sequence are input-output pairs corresponding to the algorithm's calls to the black-box oracle for h. (Recall that these later oracle calls are adaptive, i.e. each query point can depend on the answers received from previous oracle calls.)

Notation. For any oracle algorithm ALG, let $\mathcal{P}_{yes}^{ALG}$ denote the distribution over knowledge sequences induced by running ALG on a pair $(g, \mathcal{D}_g)$ randomly drawn from $\mathcal{YES}$. Similarly, let $\mathcal{P}_{no}^{ALG}$ denote the distribution over knowledge sequences induced by running ALG on a pair $(f, \mathcal{D}_f)$ randomly drawn from $\mathcal{NO}$. For $0 \leq i \leq q$ we write $\mathcal{P}_{yes,i}^{ALG}$ to denote the length-$(q+i)$ prefix of $\mathcal{P}_{yes}^{ALG}$, and similarly for $\mathcal{P}_{no,i}^{ALG}$.

We will prove Theorem 2 by showing that the statistical distance $d_{TV}(\mathcal{P}_{yes}^{T}, \mathcal{P}_{no}^{T})$ between distributions $\mathcal{P}_{yes}^{T}$ and $\mathcal{P}_{no}^{T}$ is at most $1/4$. Because of space constraints some proofs are omitted from the following presentation; all omitted proofs can be found in the appendix.

Most sequences of draws are "clean" in both the yes- and no- cases. The main result of this subsection is Corollary 1; intuitively, this corollary shows that given only q draws from the distribution and no black-box queries, it is impossible to distinguish between the yes- and no- cases with high accuracy. This is achieved via a notion of a "clean" sequence of draws from the distribution, which we now explain.

Let $S = (x^1, y_1), \ldots, (x^q, y_q)$ be a sequence of q labeled examples drawn from distribution $\mathcal{D}$, where $\mathcal{D}$ is either $\mathcal{D}_f$ for some $(f, \mathcal{D}_f) \in \mathcal{NO}$ or $\mathcal{D}_g$ for some $(g, \mathcal{D}_g) \in \mathcal{YES}$. In either case there is a corresponding set of points $a^1, b^1, c^1, \ldots, a^m, b^m, c^m$ as described in Section 3. We say that S is *clean* if S does not hit any block $1, \ldots, m$ twice, i.e. if the number of different blocks from $1, \ldots, m$ for which S contains some point a^i, b^i or c^i is exactly q. With this definition, we have the following claim and its easy corollary:

Claim. We have $\mathbf{Pr}[\mathcal{P}^T_{yes,0} \text{ is clean}] = \mathbf{Pr}[\mathcal{P}^T_{no,0} \text{ is clean}] \geq 1 - q^2/m$. Furthermore, the conditional random variables $(\mathcal{P}^T_{yes,0} \mid \mathcal{P}^T_{yes,0} \text{ is clean})$ and $(\mathcal{P}^T_{no,0} \mid \mathcal{P}^T_{no,0} \text{ is clean})$ are identically distributed.

Corollary 1. *The statistical distance $d_{TV}(\mathcal{P}^T_{yes,0}, \mathcal{P}^T_{no,0})$ is at most q^2/m.*

Eliminating foolhardy queries. Let T' denote a modified version of algorithm T which works as follows: like T, it starts out by making q draws from the distribution. Let Q be the set of all indices i such that all q draws from the distribution have the i-th bit set to 1. We say that any query string $x \in \{0,1\}^n$ that has $x_j = 0$ for some $j \in Q$ is *foolhardy*. After making its q draws from $\mathcal{D}$, algorithm T' simulates algorithm T for q black-box queries, except that for any foolhardy query that T makes, T' "fakes" the query in the following sense: it does not actually make the query but instead proceeds as T would proceed if it made the query and received the response 0.

Our goal in this subsection is to show that in both the yes- and no- cases, the executions of T and T' are statistically close. (Intuitively, this means that we can w.l.o.g. assume that the testing algorithm T does not make any foolhardy queries.) To analyze algorithm T' it will be useful to consider some other algorithms that are intermediate between T and T', which we now describe.

For each value $1 \leq k \leq q$, let U_k denote the algorithm which works as follows: U_k first makes q draws from the distribution $\mathcal{D}$, then simulates algorithm T for k queries, except that for each of the first $k-1$ queries that T makes, if the query is foolhardy then U_k "fakes" the query as described above. Let U'_k denote the algorithm which works exactly like U_k, except that if the k-th query made by U_k is foolhardy then U'_k fakes that query as well. We have the following:

Lemma 2. *For all $k \in [q]$, the statistical distance*

$$d_{TV}((\mathcal{P}^{U_k}_{yes} \mid \mathcal{P}^{U_k}_{yes,0} \textit{ is clean}), (\mathcal{P}^{U'_k}_{yes} \mid \mathcal{P}^{U'_k}_{yes,0} \textit{ is clean}))$$

is at most $2\ell m/n$, and similarly $d_{TV}((\mathcal{P}^{U_k}_{no} \mid \mathcal{P}^{U_k}_{no,0} \textit{ is clean}), (\mathcal{P}^{U'_k}_{no} \mid \mathcal{P}^{U'_k}_{no,0} \textit{ is clean}))$ is also at most $2\ell m/n$.

Now a hybrid argument using Lemma 2 lets us bound the statistical distance between the executions of T and T'.

Lemma 3. *The statistical distance $d_{TV}(\mathcal{P}^{T'}_{yes}, \mathcal{P}^T_{yes})$ is at most $2\ell mq/n + q^2/m$, and the same bound holds for $d_{TV}(\mathcal{P}^{T'}_{no}, \mathcal{P}^T_{no})$.*

Bounding the probability of finding a witness. Let T'' denote an algorithm that is a variant of T', modified as follows. T'' simulates T' except that T'' does not actually make queries on non-foolhardy strings; instead T'' simulates the answers to those queries "in the obvious way" that they should be answered if the target function were a yes-function and hence all of the draws from $\mathcal{D}$ that yielded strings with ℓ zeros were in fact b^i-type points. More precisely, assume that there are r distinct c^i-type points in the initial sequence of q draws from the distribution. Since for each c^i-type point the algorithm is given $\alpha(i)$, the algorithm "knows" r variables $x_{\alpha(i)}$ that are in the conjunction. To simulate an answer to a non-foolhardy query $x \in \{0,1\}^n$, T'' answers with 0 if any of the r $x_{\alpha(i)}$ variables are set to 0 in x, and answers with 1 otherwise. Note that consequently T'' does not actually make any black-box queries at all.

In this subsection we will show that in both the yes- and no- cases, the executions of T' and T'' are statistically close; once we have this it is not difficult to complete the proof of Theorem 2. In the yes-case these distributions are in fact identical (Lemma 4), but in the no-case these distributions are not identical; we will argue that they are close using properties of the function f' from Section 3.2.

We first address the easier yes-case:

Lemma 4. *The statistical distance $d_{TV}(\mathcal{P}^{T'}_{yes}, \mathcal{P}^{T''}_{yes})$ is zero.*

Proof. We argue that T' and T'' answer all queries in exactly the same way. Fix any $1 \le i \le q$ and let z denote the i^{th} query made by T.

If z is a foolhardy query then both T' and T'' answer z with 0. So suppose that z is not a foolhardy query. Then any 0's that z contains must be in positions from points that were sampled in the first stage. Consequently the only variables that can be set to 0 that are in the conjunction g are the $x_{\alpha(i)}$ variables from the C_i sets corresponding to the c^i points in the draws. All the other variables that were "seen" are not in the conjunction so setting them to 0 or 1 will not affect the value of $g(z)$. Therefore, $g(z)$ (and hence T''s response) is 0 if any of the $x_{\sigma(i)}$ variables are set to 0 in z, and is 1 otherwise. This is exactly how T'' answers non-foolhardy queries as well. □

To handle the no-case, we introduce the notion of a "witness" that the black-box function is a no-function.

Definition 3. *We say that a knowledge sequence contains a* witness *for $(f, \mathcal{D}_f)$ if elements $q+1, \ldots$ of the sequence (the black-box queries) contain either of the following:*

1. *A point $z \in \{0,1\}^n$ such that for some $1 \le i \le m$ for which a^i was sampled in the first q draws, the bit $z_{\alpha(i)}$ is 0 but fewer than s of the elements $j \in A_i$ have $z_j = 0$. We refer to such a point as an a-*witness for block i.
2. *A point $z \in \{0,1\}^n$ such that for some $1 \le i \le m$ for which c^i was sampled in the first q draws, z is i-special. We refer to such a point as a c-*witness for block i.

The following lemma implies that it is enough to bound the probability that $\mathcal{P}^T_{no}$ contains a witness:

Lemma 5. *The statistical distance $d_{TV}((\mathcal{P}^{T'}_{no} \mid \mathcal{P}^{T'}_{no}$ does not contain a witness and $\mathcal{P}^{T'}_{no,0}$ is clean$), (\mathcal{P}^{T''}_{no} \mid \mathcal{P}^{T''}_{no}$ does not contain a witness and $\mathcal{P}^{T''}_{no,0}$ is clean$))$ is zero.*

Proof. Claim 4.2 implies that $(\mathcal{P}^{T'}_{no,0} \mid \mathcal{P}^{T'}_{no,0}$ is clean$)$ and $(\mathcal{P}^{T''}_{no,0} \mid \mathcal{P}^{T''}_{no,0}$ is clean$)$ are identically distributed. We show that if there is no witness then T' and T'' answer all queries in exactly the same way; this gives the lemma. Fix any $1 \leq i \leq q$ and let z denote the i^{th} query.

If z is a foolhardy query, then both T' and T'' answer z with 0. So suppose that z is not a foolhardy query and not a witness. Then any 0's that z contains must be in positions from points that were sampled in the first stage.

First suppose that one of the $x_{\alpha(i)}$ variables from some c^i that was sampled is set to 0 in z. Since z is not a witness, either z has fewer than s zeros from A_i or some variable from B_i is set to zero in z. So in this case we have $f(x_i) = g_2(x_i) = 0$.

Now suppose that none of the $x_{\alpha(i)}$ variables from the c^i's that were sampled are set to 0 in z. If no variable $x_{\alpha(i)}$ from any a^i that was sampled is set to 0 in z, then clearly $f(z) = g(z) = 1$. If any variable $x_{\alpha(i)}$ from an a^i that was sampled is set to 0 in z, then since z is not a witness there must be at least s elements of A_i set to 0 and every element of B_i set to 1 for each such $x_{\alpha(i)}$. Therefore, $f(z) = 1$.

Thus $f(z)$ evaluates to 0 if any of the $x_{\sigma(i)}$ variables from the c^i's that were sampled is set to 0 and evaluates to 1 otherwise. This is exactly how T'' answers queries as well. □

Let us consider a sequence of algorithms that hybridize between T' and T'', similar to the previous section. For each value $1 \leq k \leq q$, let V_k denote the algorithm which works as follows: V_k first makes q draws from the distribution $\mathcal{D}$, then simulates algorithm T' for k queries, except that each of the first $k-1$ queries is faked (foolhardy queries are faked as described in the previous subsection, and non-foolhardy queries are faked as described at the start of this subsection). Thus algorithm V_k actually makes at most one query to the black-box oracle, the k-th one (if this is a foolhardy query then this one is faked as well). Let V'_k denote the algorithm which works exactly like V_k, except that if the k-th query made by V_k is non-foolhardy then V'_k fakes that query as well as described at the start of this subsection.

Lemma 6. *For each value $1 \leq k \leq q$, the statistical distance $d_{TV}((\mathcal{P}^{V_k}_{no} \mid \mathcal{P}^{V_k}_{no,0}$ is clean $), (\mathcal{P}^{V'_k}_{no} \mid \mathcal{P}^{V'_k}_{no,0}$ is clean$))$ is at most $\max\{\frac{qs}{\ell}, \frac{q}{2^s}\} = qs/\ell$.*

Proof. By Lemma 5, the executions of V_k and V'_k are identically distributed unless the k-th query string (which we denote z) is a witness for $(f, \mathcal{D}_f)$. Since neither V_k nor V'_k makes any black-box query prior to z, the variation distance between $\mathcal{P}^{V_k}_{no}$ and $\mathcal{P}^{V'_k}_{no}$ is at most $\mathbf{Pr}[z$ is a witness$]$ where the probability is taken over a random draw of $(f, \mathcal{D}_f)$ from $\mathcal{NO}$ conditioned on $(f, \mathcal{D}_f)$ being consistent with the q draws from the distribution and with those first q draws being clean. We bound the probability that z is a witness by considering both possibilities for z (an a-witness or a c-witness) in turn.

– We first bound the probability that z is an a-witness. So fix some $i \in [m]$ and let us suppose that a^i was sampled in the first stage of the algorithm. We will bound the probability that z is an a-witness for block i; once we have done this, a union bound over the (at most q) blocks such that a^i is sampled in the first stage gives a bound on the overall probability that z is an a-witness.

Fix any possible outcome for z. In order for z to be an a-witness for block i, it must be the case that fewer than s of the ℓ elements in A_i are set to 0 in z, but the bit $z_{\alpha(i)}$ is set to 0. For a random choice of $(f, \mathcal{D}_f)$ as described above, since we are conditioning on the q draws from the distribution being clean, the only information that these q draws reveal about the index $\alpha(i)$ is that it is some member of the set A_i. Consequently for a random $(f, \mathcal{D}_f)$ as described above, each bit in A_i is equally likely to be chosen as $\alpha(i)$, so the probability that $\alpha(i)$ is chosen to be one of the at most s bits in A_i that are set to 0 in z is at most s/ℓ. Consequently the probability that z is an a-witness for block i is at most s/ℓ, and a union bound gives that the overall probability that z is an a-witness is at most qs/ℓ.

– Now we bound the probability that z is a c-witness. Fix some $i \in [m]$ and let us suppose that c^i was sampled in the first stage of the algorithm. We will bound the probability that z is a c-witness for block i and then use a union bound as above.
Fix any possible outcome for z; let r denote the number of 0's that z has in the bit positions in C_i. In order for z to be a c-witness for block i it must be the case that z is i-special, i.e. $r \geq s$ and all r of these 0's in fact belong to A_i. For a random choice of $(f, \mathcal{D}_f)$ conditioned on being consistent with the q samples from the distribution and with those q samples being clean, the distribution over possible choices of A_i is uniform over all $\binom{2\ell}{\ell}$ possibilities for selecting a size-ℓ subset of C_i. Consequently the probability that all r 0's belong to A_i is at most

$$\frac{\binom{2\ell-r}{\ell-r}}{\binom{2l}{\ell}} = \frac{\ell(\ell-1)\cdots(\ell-r+1)}{2\ell(2\ell-1)\cdots(2\ell-r+1)} < \frac{1}{2^r} \leq \frac{1}{2^s}.$$

So the probability that z is a c-witness for block i is at most $1/2^s$, and by a union bound the overall probability that z is a c-witness is at most $q/2^s$.

So the overall probability that z is a witness is at most $\max\{\frac{qs}{\ell}, \frac{q}{2^s}\}$. Using (1) we have that the maximum is qs/ℓ, and the lemma is proved. □

Now similar to Section 4.2, a hybrid argument using Lemma 6 lets us bound the statistical distance between the executions of T' and T''. The proof of the following lemma is entirely similar to that of Lemma 3 so we omit it.

Lemma 7. *The statistical distance $d_{TV}(\mathcal{P}_{no}^{T'}, \mathcal{P}_{no}^{T''})$ is at most $q^2 s/\ell + q^2/m$.*

Putting the pieces together. At this stage, we have that T'' is an algorithm that only makes draws from the distribution and makes no queries. It follows that the statistical distance $d_{TV}(\mathcal{P}_{yes}^{T''}, \mathcal{P}_{no}^{T''})$ is at most $d_{TV}(\mathcal{P}_{yes,0}^{T}, \mathcal{P}_{no,0}^{T})$. So we can bound $d_{TV}(\mathcal{P}_{yes}^{T}, \mathcal{P}_{no}^{T})$ as follows (we write "d" in place of "d_{TV}" for brevity):

$$\begin{aligned} & d(\mathcal{P}_{yes}^{T}, \mathcal{P}_{yes}^{T'}) + d(\mathcal{P}_{yes}^{T'}, \mathcal{P}_{yes}^{T''}) + d(\mathcal{P}_{yes}^{T''}, \mathcal{P}_{no}^{T''}) + d(\mathcal{P}_{no}^{T''}, \mathcal{P}_{no}^{T'}) + d(\mathcal{P}_{no}^{T'}, \mathcal{P}_{no}^{T}) \\ \leq\ & d(\mathcal{P}_{yes}^{T}, \mathcal{P}_{yes}^{T'}) + d(\mathcal{P}_{yes}^{T'}, \mathcal{P}_{yes}^{T''}) + d(\mathcal{P}_{yes,0}^{T}, \mathcal{P}_{no,0}^{T}) + d(\mathcal{P}_{no}^{T''}, \mathcal{P}_{no}^{T'}) + d(\mathcal{P}_{no}^{T'}, \mathcal{P}_{no}^{T}) \\ \leq\ & 4q^2/m + 4q\ell m/n + q^2 s/\ell \end{aligned}$$

where the final bound follows by combining Corollary 1, Lemma 3, Lemma 4 and Lemma 7. Recalling the parameter settings $\ell = n^{2/5}(\log n)^{3/5}$, $m = (n/\log n)^{2/5}$, and $s = \log n$ from (1) and the fact that $q = \frac{1}{20}(\frac{n}{\log n})^{1/5}$, this bound is less than $1/4$. This concludes the proof of Theorem 2. $\square$

5 Extending the lower bound to conjunctions and decision lists

The construction and analysis from the previous sections easily give a lower bound for testing decision lists via the following lemma:

Lemma 8. *For any pair $(f, \mathcal{D}_f)$ in the support of $\mathcal{NO}$ and any decision list h, the function f is at least $1/6$-far from h w.r.t. $\mathcal{D}_f$.*

Proof. Fix any $(f, \mathcal{D}_f)$ in the support of $\mathcal{NO}$ and any decision list $h = (\ell_1, \beta_1)$, (ℓ_2, β_2), $\ldots$, (ℓ_k, β_k), β_{k+1}. We will show that at least one of the six points a^1, b^1, c^1, a^2, b^2, c^2 is labeled differently by h and f. Grouping all m blocks into pairs and applying the same argument to each pair gives the lemma.

Let ℓ_{a_1} be the first literal in h that is satisfied by a^1, so the value $h(a^1)$ equals β_{a_1}. Define ℓ_{b_1}, ℓ_{c_1}, ℓ_{a_2}, ℓ_{b_2}, and ℓ_{c_2} similarly. We will assume that h and f agree on all six points, i.e. that $\beta_{a_1} = \beta_{b_1} = \beta_{a_2} = \beta_{b_2} = 1$ and $\beta_{c_1} = \beta_{c_2} = 0$, and derive a contradiction.

We may suppose w.l.o.g. that $a_1 = \min\{a_1, b_1, a_2, b_2\}$. We now consider two cases depending on whether or not $c_1 < a_1$. (Note that a_1 cannot equal c_1 since $f(a^1) = 1$ but $f(c^1) = 0$.)

Suppose first that $c_1 < a_1$. No matter what literal ℓ_{c_1} is, since c^1 satisfies ℓ_{c_1} at least one of a^1, b^1 must satisfy it as well. But this means that $\min\{a_1, b_1\} \leq c_1$, which is impossible since $c_1 < a_1$ and $a_1 \leq \min\{a_1, b_1\}$.

Now suppose that $a_1 < c_1$; then it must be the case that ℓ_{a_1} is a literal "x_j" for some $j \in B_1$. (The only other possibilities are that ℓ_{a_1} is "$\overline{x}_j$" for some $j \in A_i$ or is "x_j" for some $j \in ([n] \setminus C_1)$; in either case, this would imply that $f(c^1) = 1$, which does not hold.) Since $f(c^2) = 0$ and $(c^2)_j = 1$, it must be the case that $c_2 < a_1$. But no matter what literal ℓ_{c_2} is, since c^2 satisfies it at least one of a^2, b^2 must satisfy it as well. This means that $\min\{a_2, b_2\} \leq c_2 < a_1 \leq \min\{a_2, b_2\}$, which is a contradiction. $\square$

Since monotone conjunctions are a subclass of decision lists, for every $(g, \mathcal{D}_g)$ in the support of $\mathcal{YES}$ we have that g is computed by a decision list. We thus have the obvious analogue of Lemma 1 for decision lists; together with Theorem 2, this gives the $\Omega((n/\log n)^{1/5})$ lower bound for decision lists that is claimed in Theorem 1.

Since any conjunction (not necessarily monotone) can be expressed as a decision list, we immediately have an analogue of Lemma 8 for general conjunctions. The same line of reasoning described above now gives the $\Omega((n/\log n)^{1/5})$ lower bound for general conjunctions that is claimed in Theorem 1.

References

1. N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Information and Computation*, 204:1704–1717, 2006.

2. N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. Testing low-degree polynomials over GF(2). In *Proceedings of RANDOM-APPROX*, pages 188–199, 2003.
3. N. Alon and A. Shapira. Homomorphisms in Graph Property Testing - A Survey. Topics in Discrete Mathematics (to appear), available at http://www.math.tau.ac.il/ãsafico/nesetril.pdf, 2007.
4. M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comp. Sys. Sci.*, 47:549–595, 1993. Earlier version in STOC'90.
5. I. Diakonikolas, H. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. Servedio, and A. Wan. Testing for concise representations. Submitted for publication, 2007.
6. Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonocity. In *Proceedings of RANDOM*, pages 97–108, 1999.
7. E. Fischer. The art of uninformed decisions: A primer to property testing. *Computational Complexity Column of The Bulletin of the European Association for Theoretical Computer Science*, 75:97–126, 2001.
8. E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science*, pages 103–112, 2002.
9. E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samrodnitsky. Monotonicity testing over general poset domains. In *Proc. 34th Annual ACM Symposium on the Theory of Computing*, pages 474–483, 2002.
10. O. Goldreich. Combinatorial property testing – a survey. In "Randomized Methods in Algorithms Design", AMS-DIMACS, 45–61, 1998.
11. O. Goldreich, S. Goldwaser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.
12. O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samordinsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.
13. S. Halevy and E. Kushilevitz. Distribution-Free Property Testing. In *Proceedings of the Seventh International Workshop on Randomization and Computation*, pages 302–317, 2003.
14. S. Halevy and E. Kushilevitz. Distribution-Free Connectivity Testing. In *Proceedings of the Eighth International Workshop on Randomization and Computation*, pages 393–404, 2004.
15. S. Halevy and E. Kushilevitz. A lower bound for distribution-free monotonicity testing. In *Proceedings of the Ninth International Workshop on Randomization and Computation*, pages 330–341, 2005.
16. M. Kearns and U. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, 1994.
17. K. Matulef, R. O'Donnell, R. Rubinfeld, and R. Servedio. Testing Halfspaces. Manuscript, 2007.
18. M. Parnas, D. Ron, and A. Samorodnitsky. Testing basic boolean formulae. *SIAM J. Disc. Math.*, 16:20–46, 2002.
19. D. Ron. Property testing (a tutorial). In "Handbook of Randomized Computing, Volume II", S. Rajasekaran and P. M. Pardalos and J. H. Reif and J. D. P. Rolim, editors, Kluwer, 2001.
20. R. Rubinfeld. Sublinear time algorithms. available at http://theory.csail.mit.edu/˜ronitt/papers/icm.ps, 2006.
21. R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. on Comput.*, 25:252–271, 1996.
22. G. Turán. Lower bounds for PAC learning with queries. In *COLT '93: Proc. 6th Annual Conference on Computational Learning Theory*, pages 384–391, 2002.

A Proofs from Section 4.2

A.1 Proof of Claim 4.2:

We first show that $\mathbf{Pr}[\mathcal{P}^T_{yes,0} \text{ is clean}] = \mathbf{Pr}[\mathcal{P}^T_{no,0} \text{ is clean}] \geq 1 - q^2/m$. Fix any $(g, \mathcal{D}_g)$ in the support of $\mathcal{YES}$, and consider the outcomes of $\mathcal{P}^T_{yes,0}$ corresponding to this $(g, \mathcal{D}_g)$ being drawn from $\mathcal{YES}$. Since each independent draw from $\mathcal{D}_g$ hits each block $1, \ldots, m$ with probability $1/m$, the probability that $\mathcal{P}^T_{yes,0}$ is clean is $\prod_{i=1}^{q} \left(1 - \frac{i-1}{m}\right) \geq 1 - q^2/m$. The same argument shows that $\mathbf{Pr}[\mathcal{P}^T_{no,0} \text{ is clean}]$ also equals $\prod_{i=1}^{q} \left(1 - \frac{i-1}{m}\right)$.

Now we show that the conditional random variables are identically distributed. It is not difficult to see that for any $0 \leq j \leq q-1$, given any particular length-j prefix of $\mathcal{P}^T_{yes,0}$, conditioned on $\mathcal{P}^T_{yes,0}$ being clean, the $(j+1)$-st element of $\mathcal{P}^T_{yes,0}$ has

– a $2/3$ chance of being a triple $(x, 1, 0)$ where $x \in \{0,1\}^n$ has ℓ zeros and the locations of the ℓ zeros are selected uniformly at random (without replacement) from the set of those bit positions that had value 1 in all j of the previous draws;
– a $1/3$ chance of being a triple $(x, 0, \alpha)$ where x has 2ℓ zeros, the locations of the 2ℓ zeros are selected uniformly at random (without replacement) from the same set of bit positions described above, and α is an index drawn uniformly at random from the indices of the 2ℓ zeros in x.

It is also not difficult to see that given any particular length-j prefix of $\mathcal{P}^T_{no,0}$, conditioned on $\mathcal{P}^T_{no,0}$ being clean, the $(j+1)$-st element of $\mathcal{P}^T_{no,0}$ is distributed in the exact same way. This proves Claim 4.2. □

A.2 Proof of Corollary 1:

We can express the statistical distance between $\mathcal{P}^T_{yes,0}$ and $\mathcal{P}^T_{no,0}$ as

$$\frac{1}{2} \sum_{\zeta} \Big| \mathbf{Pr}[(\mathcal{P}^T_{yes,0} = \zeta) \,\&\, (\mathcal{P}^T_{yes,0} \text{ is clean})] + \mathbf{Pr}[(\mathcal{P}^T_{yes,0} = \zeta) \,\&\, (\mathcal{P}^T_{yes,0} \text{ not clean})] - \mathbf{Pr}[\mathcal{P}^T_{no,0} = \zeta) \,\&\, (\mathcal{P}^T_{no,0} \text{ is clean})] - \mathbf{Pr}[(\mathcal{P}^T_{no,0} = \zeta) \,\&\, (\mathcal{P}^T_{no,0} \text{ not clean})] \Big| .$$

By parts (1) and (2) of the claim, we have that $\mathbf{Pr}[(\mathcal{P}^T_{yes,0} = \zeta) \,\&\, (\mathcal{P}^T_{yes,0} \text{ is clean})]$ equals $\mathbf{Pr}[(\mathcal{P}^T_{no,0} = \zeta) \,\&\, (\mathcal{P}^T_{no,0} \text{ is clean})]$ for all ζ. Thus we can reexpress the statistical distance as

$$\frac{1}{2} \sum_{\zeta} \Big| \mathbf{Pr}[(\mathcal{P}^T_{yes,0} = \zeta) \,\&\, (\mathcal{P}^T_{yes,0} \text{ not clean})] - \mathbf{Pr}[\mathcal{P}^T_{no,0} = \zeta) \,\&\, (\mathcal{P}^T_{no,0} \text{ not clean})] \Big| .$$

This is at most $\frac{1}{2}(\mathbf{Pr}[\mathcal{P}^T_{yes,0} \text{ not clean}] + \mathbf{Pr}[\mathcal{P}^T_{no,0} \text{ not clean}])$ which is at most q^2/m by part (1) of the claim. □

A.3 Proof of Lemma 2:

We consider the yes-case; the no-case follows by an essentially identical argument.

The executions of U_k and U'_k are identically distributed unless the k-th query string (which we denote z) is foolhardy and the black-box function g has $g(z) = 1$. Consequently the variation distance $d_{TV}(\mathcal{P}_{yes}^{U_k}, \mathcal{P}_{yes}^{U'_k})$ is at most

$$\mathbf{Pr}[(z \text{ is foolhardy}) \;\&\; (g(z) = 1)] \leq \mathbf{Pr}[(g(z) = 1) \mid (z \text{ is foolhardy})],$$

where the probabilities are taken over a random draw of $(g, \mathcal{D}_g)$ from $\mathcal{YES}$ conditioned on $(g, \mathcal{D}_g)$ being consistent with the q draws from the distribution and with the first $k-1$ queries, and with the q draws from the distribution being clean.

Since the first $k-1$ queries do not involve any variables in Q (because foolhardy queries are faked for the first $k-1$ queries) and our analysis will only concern variables in Q, to analyze this conditional probability it is enough to consider $(g, \mathcal{D}_g)$ drawn from $\mathcal{YES}$ conditioned on $(g, \mathcal{D}_g)$ being consistent with the q draws from the distribution and with these q draws being clean. Suppose that these draws from the distribution yield r c^i-type points (each with 2ℓ zeros) and $(q-r)$ b^i-type points (each with ℓ zeros). Then after these draws, the algorithm "knows" $(r+q)\ell$ elements of R. Let Z denote the set of these $(r+q)\ell$ elements of R. The set R also contains $2\ell m - (r+q)\ell$ other "unknown" variables from among the $|Q| = n - (r+q)\ell$ variables in $[n] \setminus Z$.

We would like to find the probability, over random $(g, \mathcal{D}_g)$ drawn from $\mathcal{YES}$ consistent with the draws from the distribution, that $g(z) = 1$ given that z is foolhardy. Since z is foolhardy there must be at least one index $j \in [n] \setminus Z$ such that $z_j = 0$. So the desired probability is at most the probability that j belongs to R, since if $j \notin R$ the conjunction g_1 will evaluate to 0 on z. For a random $(g, \mathcal{D}_g)$ that is consistent with the draws from the distribution, the remaining $2\ell m - (r+q)\ell$ elements of $R \setminus Z$ are chosen randomly from the $n - (r+q)\ell$ elements of $[n] - Z$. Consequently the probability that j belongs to $R \setminus Z$ is $\frac{2\ell m - (r+q)\ell}{n - (r+q)\ell} \leq \frac{2\ell m}{n}$, and the lemma is proved. $\square$

A.4 Proof of Lemma 3:

We prove the yes-case; the no-case follows by an identical argument.

By Claim 4.2, at the cost of q^2/m in $d_{TV}(\mathcal{P}_{yes}^{T'}, \mathcal{P}_{yes}^{T})$ we may assume that the draws from the distribution are clean. So we henceforth in the proof always condition on the draws from the distribution being clean, and we will bound $d_{TV}(\mathcal{P}_{yes}^{T'}, \mathcal{P}_{yes}^{T})$ by $2\ell mq/n$ under this conditioning on each argument to d_{TV}.

We use induction on i to show that $d_{TV}(\mathcal{P}_{yes,i}^{T}, \mathcal{P}_{yes,i}^{T'})$ is at most $2\ell lmi/n$ for all i. Once we have this, taking $i = q$ and recalling that $\mathcal{P}_{yes,q}^{T} = \mathcal{P}_{yes}^{T}$ and $\mathcal{P}_{yes,q}^{T'} = \mathcal{P}_{yes}^{T'}$ gives the desired bound.

The base case $i = 0$ is clear since in this case no black-box queries are made by either T or T'.

For the induction step we assume that $d_{TV}(\mathcal{P}_{yes,i}^{T}, \mathcal{P}_{yes,i}^{T'}) \leq 2\ell mi/n$, and we will show that $d_{TV}(\mathcal{P}_{yes,i+1}^{T}, \mathcal{P}_{yes,i+1}^{T'}) \leq 2\ell m(i+1)/n$. We first note that the random variables $\mathcal{P}_{yes,i+1}^{T'}$ and $\mathcal{P}_{yes}^{U'_{i+1}}$ are identically distributed, i.e. they have statistical distance zero. Lemma 2 now implies that $d_{TV}(\mathcal{P}^{T'}_{yes,i+1}, \mathcal{P}^{U_{i+1}}_{yes})$ is at most $2\ell m/n$. Since

$$d_{TV}(\mathcal{P}^{T'}_{yes,i+1}, \mathcal{P}^{T}_{yes,i+1}) \le d_{TV}(\mathcal{P}^{T'}_{yes,i+1}, \mathcal{P}^{U_{i+1}}_{yes}) + d_{TV}(\mathcal{P}^{U_{i+1}}_{yes}, \mathcal{P}^{T}_{yes,i+1}),$$

it is enough to bound $d_{TV}(\mathcal{P}^{U_{i+1}}_{yes}, \mathcal{P}^{T}_{yes,i+1})$ by $2\ell mi/n$. But since the first $q+i$ elements of $\mathcal{P}^{U_{i+1}}_{yes}$ are distributed according to $\mathcal{P}^{T'}_{i}$ (and the last element is obtained by performing the i-th query of T), the bound $d_{TV}(\mathcal{P}^{U_{i+1}}_{yes}, \mathcal{P}^{T}_{yes,i+1}) \le 2\ell mi/n$ follows from the induction hypothesis. This concludes the proof. □

B The two distributions for linear threshold functions

Now we would like to prove a lower bound for distribution-free testing of the class LTF. The construction from Section 3 is not suited for a lower bound on LTF (observe that for any $(f, \mathcal{D}_f)$ in the support of $\mathcal{NO}$ the function f is 0-far from the linear threshold function $x_1 + \cdots + \cdots x_n \ge n - 3\ell/2$ with respect to $\mathcal{D}_f$), so we need a different approach.

In the rest of this section we define two distributions $\mathcal{YES}$ and $\mathcal{NO}$ over pairs $(h, \mathcal{D})$ and prove that these distributions have the following properties:

1. For every pair $(g, \mathcal{D}_g)$ in the support of $\mathcal{YES}$, the function g is a linear threshold function;
2. For every pair $(f, \mathcal{D}_f)$ in the support of $\mathcal{NO}$, the function f is $1/4$-far from LTF with respect to $\mathcal{D}_f$ (and hence any tester for LTF must accept every such pair with probability at most $1/3$).

In Section C we use these distributions to prove a lower bound for LTF.

Before giving the precise construction, here is a very rough first intuition for how it works. Recall that in the earlier construction, we relied on the fact that no monotone conjunction h can satisfy $h(1,0) = h(0,1) = 1$ but $h(0,0) = 0$. For linear threshold functions, we will instead rely on the fact that no linear threshold function can satisfy $h(0,0) = h(1,1) = 0$ but $h(1,0) = h(0,1) = 1$.

B.1 The $\mathcal{YES}$ distribution.

As in Sections 3 and 4 our constructions are parameterized by values ℓ, m and s that are set according to (1). A draw from the distribution $\mathcal{YES}$ over $(g, \mathcal{D}_g)$ pairs is obtained as follows:

- As before let $R \subset [n]$ be a set of size $2\ell m$ selected uniformly at random.
- As before, randomly partition the set R into $2m$ subsets $A_1, B_1, \ldots, A_m, B_m$, each of size ℓ. Let $C_i = A_i \cup B_i$ and let a^i, b^i, c^i be defined as before. As before, for each $i = 1, \ldots, m$ choose $\alpha(i)$ to be a random element of the set A_i.
- The distribution $\mathcal{D}_g$ puts $1/4$ weight on the point 1^n, and puts weight $1/(2m)$ on b^i and $1/(4m)$ on c^i for all $i = 1, \ldots, m$.

– The function g is defined as follows: $g(x)$ equals 1 if $u(x) \geq \theta$ and equals 0 if $u(x) < \theta$, where

$$u(x) \stackrel{\text{def}}{=} 10n^2 \sum_{j\in([n]\setminus R)} x_j + 5n \sum_{i=1}^{m} x_{\alpha(i)} - \sum_{i=1}^{m} \sum_{k\in C_i, k\neq\alpha(i)} x_k, \qquad (2)$$

$$\theta \stackrel{\text{def}}{=} 10n^2(n - 2\ell m) + 5nm - m(2\ell - 1) + s. \qquad (3)$$

An equivalent way to define g is that $g(x) = 1$ if and only if all three of the following conditions hold:

1. $x_j = 1$ for all $j \in ([n] \setminus R)$;
2. $x_j = 1$ for all $j = \alpha(1), \dots, \alpha(m)$; and
3. $\sum_{i=1}^{m} \sum_{k\neq\alpha(i)} x_k \leq m(2\ell - 1) - s$.

Fix any $(g, \mathcal{D}_g)$ in the support of $\mathcal{YES}$. It is clear that g is a linear threshold function. It is straightforward to check that $u(1^n) = 10n^2(n - 2\ell m) + 5nm - m(2\ell - 1)$, $u(c^i) = 10n^2(n - 2\ell m) + 5n(m - 1) - (m - 1)(2\ell - 1)$, and $u(b^i) = 10n^2(n - 2\ell m) + 5nm - m(2\ell - 1) + \ell$, and consequently we have $g(1^n) = g(c^i) = 0$, $g(b^i) = 1$.

B.2 The $\mathcal{NO}$ distribution.

A draw from the distribution $\mathcal{NO}$ of $(f, \mathcal{D}_f)$ pairs is obtained as follows:

– As in the yes-case, let $R \subset [n]$ be a randomly selected set of size $2\ell m$, and randomly partition the set R into $2m$ subsets $A_1, B_1, \dots, A_m, B_m$, each of size ℓ. The points a^i, b^i, c^i, sets A_i, B_i, C_i, and indices $\alpha(i)$ are defined as in the yes-case. The distribution $\mathcal{D}_f$ puts weight $1/4$ on 1^n, puts $1/4$ weight uniformly over the m points $c^1, \dots, c^m$, and puts $1/2$ weight uniformly over the $2m$ points $a^1, b^1, \dots, a^m, b^m$.

– The function f is defined as follows: $f(x)$ equals 1 if $v(x) \geq \theta$ and equals 0 if $v(x) < \theta$. Here θ is defined as in (3) and $v(x)$ is defined as follows: given input x, let $J(x) \subseteq [m]$ be the set of those i such that x is i-special as defined in Section 3.2 (i.e. the i-th block of x has no zeros in B^i but has at least s zeros in A_i.) The function $v(x)$ is

$$\begin{aligned} v(x) \stackrel{\text{def}}{=} 10n^2 \sum_{j\in([n]\setminus R)} x_j + 5n \left(|J(x)| + \sum_{i\in([m]\setminus J)} x_{\alpha(i)} \right) \\ -|J(x)|(\ell - 1) - \sum_{i\in J} \sum_{k\in A_i} x_k - \sum_{i\in([m]\setminus J)} \sum_{k\in C_i, k\neq\alpha(i)} x_k. \qquad (4) \end{aligned}$$

Here is some intuition for the definition of f. Suppose that testing algorithm T manages to query an input string x which has $x_j = 1$ for all $j \in B_i$ but also has at least d bits in A_i set to 0. Then as we will see, it must be the case that the algorithm actually drew the point a^i in its sample from $\mathcal{D}_f$. So in order for T to be "fooled" into thinking that the function is a $\mathcal{YES}$ function, we want the contribution from the bits of A_i and B_i for this input to "look like" the function is a $\mathcal{YES}$ function for which the point a^i that was drawn from $\mathcal{D}_f$ is actually a point b^i drawn from $\mathcal{D}_g$. This is the rationale behind the definition of f; instead of computing $u(x)$ and comparing it with θ, we compute $v(x)$, which reverses the role of A_i and B_i bits on those blocks.

It is easy to see that in both the yes-case and the no-case, any black-box query that sets any variable in $[n] \setminus R$ to 0 will give a 0 response. As in the earlier construction, intuitively this will let us assume that any testing algorithm that has obtained strings $z^1, \ldots, z^q$ from the distribution $\mathcal{D}$ never queries any string x that has any bit x_i set to 0 that was set to 1 in all of $z^1, \ldots, z^q$.

Finally, it is easy to check that for any $(f, \mathcal{D}_f)$ drawn from $\mathcal{NO}$, we have $v(1^n) = 10n^2(n - 2\ell m) + 5nm - m(2\ell - 1)$, $v(c^i) = 10n^2(n - 2\ell m) + 5n(m-1) - (m-1)(2\ell - 1)$, and $v(a^i) = v(b^i) = 10n^2(n - 2\ell m) + 5nm - m(2\ell - 1) + \ell$. (Note that these values on 1^n, c^i and b^i are the same that the corresponding functions $u(x)$ would take in the yes-case.) Thus we have $f(1^n) = f(c^i) = 0$ and $f(a^i) = f(b^i) = 1$ for each $i = 1, \ldots, m$. It is easy to see that any linear threshold function must disagree with f on at least one of the four points $1^n, a^i, b^i, c^i$ for each i. Consequently f is at least $1/4$-far from any linear threshold function with respect to $\mathcal{D}_f$.

Thus we have established properties (1) and (2) stated at the beginning of this section. These yield:

Lemma 9. *Any distribution-free tester for LTF that is run with distance parameter $\epsilon = 1/4$ must accept a random pair $(g, \mathcal{D}_g)$ drawn from $\mathcal{YES}$ with probability at least $2/3$, and must accept a random pair $(f, \mathcal{D}_f)$ drawn from $\mathcal{NO}$ with probability at most $1/3$.*

C A lower bound for linear threshold functions

The basic approach is similar to that of Section 4, and indeed several ingredients from the earlier proof can be directly reused; we focus our discussion on the points where the approaches differ. We shall prove the following:

Theorem 3. *Let $q \stackrel{\text{def}}{=} \frac{1}{20}(\frac{n}{\log n})^{1/5}$. Let T be any probabilistic oracle algorithm that, given a pair $(h, \mathcal{D})$, makes at most q black-box queries to h and samples $\mathcal{D}$ at most q times. Then we have*

$$\left|\mathbf{Pr}_{(g,\mathcal{D}_g)\sim\mathcal{YES}}[T^{g,\mathcal{D}_g} = \textit{Accept}] - \mathbf{Pr}_{(f,\mathcal{D}_f)\sim\mathcal{NO}}[T^{f,\mathcal{D}_f} = \textit{Accept}]\right| \le \frac{1}{4}.$$

Note that this statement is identical to Theorem 2, but here the $\mathcal{YES}$ and $\mathcal{NO}$ distributions refer to the distributions defined in Section B.

As in Section 4.2, let T be any fixed oracle algorithm that makes exactly q draws from the distribution and then makes exactly q black-box queries. We again assume that T is given "extra information" as described earlier when it draws c^i-type examples from the distribution. Our definition of a knowledge sequence and of a "clean" sequence of draws are the same as before.

The following easy claim is an analogue of Claim 4.2:

Claim. We have $\mathbf{Pr}[\mathcal{P}^T_{yes,0}$ is clean$] = \mathbf{Pr}[\mathcal{P}^T_{no,0}$ is clean$] \geq 1 - q^2/m$. Furthermore, the conditional random variables $(\mathcal{P}^T_{yes,0} \mid \mathcal{P}^T_{yes,0}$ is clean) and $(\mathcal{P}^T_{no,0} \mid \mathcal{P}^T_{no,0}$ is clean) are identically distributed.

Proof. We first show that $\mathbf{Pr}[\mathcal{P}^T_{yes,0}$ is clean$] = \mathbf{Pr}[\mathcal{P}^T_{no,0}$ is clean$] \geq 1 - q^2/m$. Fix any $(g, \mathcal{D}_g)$ in the support of $\mathcal{YES}$, and consider the outcomes of $\mathcal{P}^T_{yes,0}$ corresponding to this $(g, \mathcal{D}_g)$ being drawn from $\mathcal{YES}$. Since each independent draw from $\mathcal{D}_g$ hits each block $1, \dots, m$ with probability $3/4m$, the probability that $\mathcal{P}^T_{yes,0}$ is clean is $\prod_{i=1}^q \left(1 - \frac{3(i-1)}{4m}\right) \geq 1 - 3q^2/4m \geq 1 - q^2/m$. The same argument shows that $\mathbf{Pr}[\mathcal{P}^T_{no,0}$ is clean$]$ also equals $\prod_{i=1}^q \left(1 - \frac{3(i-1)}{4m}\right)$.

Now we show that the conditional random variables are identically distributed. It is not difficult to see that for any $0 \leq j \leq q-1$, given any particular length-j prefix of $\mathcal{P}^T_{yes,0}$, conditioned on $\mathcal{P}^T_{yes,0}$ being clean, the $(j+1)$-st element of $\mathcal{P}^T_{yes,0}$ has

– a $1/4$ chance of being the triple $(1^n, 0, 0)$;
– a $1/2$ chance of being a triple $(x, 1, 0)$ where $x \in \{0,1\}^n$ has ℓ zeros and the locations of the ℓ zeros are selected uniformly at random (without replacement) from the set of those bit positions that had value 1 in all j of the previous draws;
– a $1/4$ chance of being a triple $(x, 0, \alpha)$ where x has 2ℓ zeros, the locations of the 2ℓ zeros are selected uniformly at random (without replacement) from the same set of bit positions described above, and α is an index drawn uniformly at random from the indices of the 2ℓ zeros in x.

It is also not difficult to see that given any particular length-j prefix of $\mathcal{P}^T_{no,0}$, conditioned on $\mathcal{P}^T_{no,0}$ being clean, the $(j+1)$-st element of $\mathcal{P}^T_{no,0}$ is distributed in the exact same way. This proves the lemma. □

The proof of the following corollary is identical to the proof of Corollary 1:

Corollary 2. *The statistical distance $d_{TV}(\mathcal{P}^T_{yes,0}, \mathcal{P}^T_{no,0})$ is at most q^2/m.*

Foolhardy queries can be handled just as before. Let the algorithms T', U_k and U'_k be defined precisely as in Section 4.2. The arguments of subsection 4.2 immediately yield:

Lemma 10. *For all $k \in [q]$, the statistical distance*

$$d_{TV}((\mathcal{P}^{U_k}_{yes} \mid \mathcal{P}^{U_k}_{yes,0} \textit{ is clean}), (\mathcal{P}^{U'_k}_{yes} \mid \mathcal{P}^{U'_k}_{yes,0} \textit{ is clean}))$$

is at most $2\ell m/n$, and similarly $d_{TV}((\mathcal{P}^{U_k}_{no} \mid \mathcal{P}^{U_k}_{no,0}$ is clean$), (\mathcal{P}^{U'_k}_{no} \mid \mathcal{P}^{U'_k}_{no,0}$ is clean$))$ is also at most $2\ell m/n$.

Lemma 11. *The statistical distance $d_{TV}(\mathcal{P}^{T'}_{yes}, \mathcal{P}^T_{yes})$ is at most $2\ell mq/n + q^2/m$, and the same bound holds for $d_{TV}(\mathcal{P}^{T'}_{no}, \mathcal{P}^T_{no})$.*

As we now describe, the details of how non-foolhardy queries are handled are different from Section 4.2.

Let T'' denote an algorithm that is a variant of T', modified as follows. T'' simulates T' except that T'' does not actually make queries on non-foolhardy strings; instead T'' simulates the answers to those queries "in the obvious way" that they should be answered if the target function were a yes-function and hence all of the draws from $\mathcal{D}$ that yielded strings with ℓ zeros were in fact b^i-type points. More precisely, assume that there are r distinct c^i-type points in the initial sequence of q draws from the distribution. Since for each c^i-type point the algorithm is given $\alpha(i)$, the algorithm "knows" r variables $x_{\alpha(i)}$ that are in the conjunction. To simulate an answer to a non-foolhardy query $z \in \{0,1\}^n$, T'' computes

$$u'(z) = 10n^2(n - 2\ell m) + 5n(m - |I'|) - m(2\ell - 1) + |K \setminus I'|$$

where:

– K is the set of all variables x_i set to 0 in z, and
– I' is the set of "known" $x_{\alpha(i)}$ variables that are set to 0 in z

and answers 1 if $u'(z) \geq \theta$, and answers 0 otherwise.

Lemma 12. *The statistical distance $d_{TV}(\mathcal{P}^{T'}_{yes}, \mathcal{P}^{T''}_{yes})$ is zero.*

Proof. We argue that T' and T'' answer all queries in exactly the same way. Fix any $1 \leq i \leq q$ and let z denote the ith query made by T.

If z is a foolhardy query then both T' and T'' answer z with 0. So suppose that z is not a foolhardy query. By inspection of (2), we can reexpress $u(z)$ as

$$u(z) = 10n^2(n - 2\ell m) + 5n(m - |I|) - m(2\ell - 1) + |K \setminus I|$$

where:

– K is the set of all variables x_i set to 0 in z, and
– I is the set of $x_{\alpha(i)}$ variables that are set to 0 in z

and $g(z)$ equals 1 if $u(z) \geq \theta$ and equals 0 if $u(z) < \theta$.

Since z is not foolhardy, the only zeros in z must be in positions from points that were sampled in the first stage. Consequently the only $x_{\alpha(i)}$ variables in I are the $x_{\alpha(i)}$ variables set to 0 in z from the C_i sets corresponding to the c^i points in the draws (these are the "known" $x_{\alpha(i)}$ variables). So $I = I'$ and $K \setminus I = K \setminus I'$.

Therefore, $u(z) = u'(z)$ and hence T''s response is 0 if $u'(z) < \theta$, and is 1 otherwise. This is exactly how T'' answers non-foolhardy queries as well. □

We define witnesses in exactly the same way as before:

Definition 4. *We say that a knowledge sequence contains a* witness *for $(f, \mathcal{D}_f)$ if elements $q+1, \ldots$ of the sequence (the black-box queries) contain either of the following:*

1. *A point $z \in \{0,1\}^n$ such that for some $1 \leq i \leq m$ for which a^i was sampled in the first q draws, the bit $z_{\alpha(i)}$ is 0 but fewer than s of the elements $j \in A_i$ have $z_j = 0$. We refer to such a point as an a-*witness for block i.

2. *A point $z \in \{0,1\}^n$ such that for some $1 \le i \le m$ for which c^i was sampled in the first q draws, z is i-special. We refer to such a point as a c-*witness for block i.

The following lemma is analogous to Lemma 5; it makes essential use of the way our no-functions f are defined in Section B.2.

Lemma 13. *The statistical distance $d_{TV}((\mathcal{P}_{no}^{T'} \mid \mathcal{P}_{no}^{T'}$ does not contain a witness and $\mathcal{P}_{no,0}^{T'}$ is clean$), (\mathcal{P}_{no}^{T''} \mid \mathcal{P}_{no}^{T''}$ does not contain a witness and $\mathcal{P}_{no,0}^{T''}$ is clean$))$ is zero.*

Proof. As in the proof of Lemma 5, we show that if there is no witness then T' and T'' answer all queries in exactly the same way. Fix any $1 \le i \le q$ and let z denote the ith query.

If z is a foolhardy query then both T' and T'' answer z with 0. So suppose that z is not a foolhardy query and not a witness. By inspection of (4), for any non-foolhardy point z we can express $v(z)$ as

$$v(z) = 10n^2(n - 2\ell m) + 5n(m - |L|) - m(2\ell - 1) + |K \setminus L|$$

where

– K is the set of all variables x_i set to 0 in z and
– L is the set of $x_{\alpha(i)}$ variables set to 0 in z such that $i \notin J(z)$ (i.e. z is not i-special).

Recall that:

$$u'(z) = 10n^2(n - 2\ell m) + 5n(m - |I'|) - m(2\ell - 1) + |K \setminus I'|$$

where

– K is the set of all variables x_i set to 0 in z and
– I' is the set of "known" $x_{\alpha(i)}$ variables that are set to 0 in z.

We will show that if z is not a witness and not foolhardy then $L = I'$. This implies that $v(z) = u'(z)$, so T' and T'' will respond to such queries in exactly the same way.

First we show that $I' \subseteq L$. Fix any $x_{\alpha(i)}$ that belongs to I'; such a variable is set to 0 in z and is "known," so c^i must have been sampled in the first stage. Since z is not a witness, z must not be i-special, i.e. $i \notin J(z)$; so $x_{\alpha(i)}$ must belong to L.

Next we show that $L \subseteq I'$. Fix any $x_{\alpha(i)}$ that belongs to L; such a variable is set to 0 in z and $i \notin J(z)$. This means z is not i-special, so z either has a zero from B_i or fewer than s zeros from A_i. Since the initial draws from $\mathcal{D}$ were clean and z is not foolhardy, it cannot be the case that a^i was drawn in the sample, for if it were drawn then z could not have a zero from B_i and also could not have fewer than s zeros from A_i (since if it had fewer than s zeros from A_i then z would be an a-witness for block i, which contradicts the fact that z is not a witness). Since a^i was not drawn in the sample but the bit $\alpha(i)$ is set to 0 in z and z is not foolhardy, it must be the case that c^i was drawn in the sample. But this means that $x_{\alpha(i)}$ is "known," and consequently $x_{\alpha(i)}$ belongs to I.

So we have shown that $I' = L$, and the lemma is proved. □

From this point on the rest of the argument from Section 4.2 can be used without modification. We define hybrid algorithms V_k, V_k' exactly as in Section 4.2, and the exact proof of Lemma 6 now yields:

Lemma 14. *For each value $1 \leq k \leq q$, the statistical distance $d_{TV}((\mathcal{P}_{no}^{V_k} \mid \mathcal{P}_{no,0}^{V_k}$ is clean $), (\mathcal{P}_{no}^{V_k'} \mid \mathcal{P}_{no,0}^{V_k'}$ is clean$))$ is at most $\max\{\frac{qs}{\ell}, \frac{q}{2^s}\} = qs/\ell$.*

Exactly as in Section 4.2, we obtain:

Lemma 15. *The statistical distance $d_{TV}(\mathcal{P}_{no}^{T'}, \mathcal{P}_{no}^{T''})$ is at most $q^2 s/\ell + q^2/m$.*

With all the pieces in place, the arguments from Section 4.2 go through unchanged to complete the proof of Theorem 3, and we are done.