Testing for Concise Representations*

Ilias Diakonikolas† Columbia University ilias@cs.columbia.edu
Homin K. Lee‡ Columbia University homin@cs.columbia.edu
Kevin Matulef§ MIT matulef@mit.edu
Krzysztof Onak¶ MIT konak@mit.edu
Ronitt Rubinfeld‖ MIT ronitt@theory.csail.mit.edu
Rocco A. Servedio** Columbia University rocco@cs.columbia.edu
Andrew Wan†† Columbia University atw12@columbia.edu

Abstract

We describe a general method for testing whether a function on n input variables has a concise representation. The approach combines ideas from the junta test of Fischer et al. [6] with ideas from learning theory, and yields property testers that make poly(s/ϵ) queries (independent of n) for Boolean function classes such as s-term DNF formulas (answering a question posed by Parnas et al. [12]), size-s decision trees, size-s Boolean formulas, and size-s Boolean circuits.

The method can be applied to non-Boolean valued function classes as well. This is achieved via a generalization of the notion of variation *from Fischer et al. to non-Boolean functions. Using this generalization we extend the original junta test of Fischer et al. to work for non-Boolean functions, and give poly(s/ϵ)-query testing algorithms for non-Boolean valued function classes such as size-s algebraic circuits and s-sparse polynomials over finite fields.*

We also prove an $\tilde{\Omega}(\sqrt{s})$ query lower bound for nonadaptively testing s-sparse polynomials over finite fields of constant size. This shows that in some instances, our general method yields a property tester with query complexity that is optimal (for nonadaptive algorithms) up to a polynomial factor.

1. Introduction

Suppose you are given black-box access to a program computing an unknown function. You would like to gain some understanding of the program by querying it as few times as possible. A natural first question is whether the program has some sort of concise representation: is it representable by a small decision tree? a small DNF formula, Boolean circuit, or algebraic circuit? a sparse polynomial?

In this paper we study the problem of testing whether a function has a concise representation for various different types of representations, including those mentioned above. We work in the standard model of *property testing*. Namely, we assume that we have black-box query access to an unknown function $f : \Omega^n \to X$, and we are interested in algorithms that accept any function which has a concise representation of the desired type and reject any function which is ϵ-far from having such a representation (i.e. for every function f' which has such a representation, f and f' disagree on at least an ϵ fraction of inputs). As is standard in property testing, we assume that queries to the function are the limiting resource (rather than computation time), and we would like to obtain algorithms whose query complexity is independent of n, the number of inputs to the function.

Previous work on testing function classes. There has been considerable research on testing functions for various types of representations. Our work is most directly motivated by the paper of Parnas *et al.* [12], who gave algorithms for testing whether Boolean functions $f : \{0,1\}^n \to \{0,1\}$ have certain very simple representations as Boolean formulae. They gave an $O(1/\epsilon)$-query algorithm for testing whether f is a single Boolean literal or a Boolean conjunction, and an $\tilde{O}(s^2/\epsilon)$-query algorithm for testing whether f is an s-term monotone DNF. Parnas *et al.* posed as an open question whether a similar testing result can be obtained for the broader

*A full version of the paper is available online.

†Supported in part by NSF grant CCF-04-30946 and an Alexander S. Onassis Foundation Fellowship.
‡Supported in part by NSF award CCF-0347282 and by NSF award CCF-0523664.
§Supported in part by an NSF graduate fellowship.
¶Supported in part by NSF grant 0514771.
‖Supported in part by NSF grant 0514771.
**Supported in part by NSF award CCF-0347282, by NSF award CCF-0523664, and by a Sloan Foundation Fellowship.
††Supported in part by NSF award CCF-0347282 and by NSF award CCF-0523664.

class of general (non-monotone) s-term DNF formulas.

Other closely related results include the following: An $O(1/\epsilon)$-query algorithm for testing whether a function can be represented as a linear form over a finite field is given in Blum *et al.* [2]. This algorithm was subsequently generalized in several works to test whether f can be represented as a low-degree polynomial. In particular, [1, 8, 9] consider the case when f is defined over a small finite field. Fischer *et al.* [6] gave an algorithm to test whether a Boolean function $f : \Omega^n \to \{0, 1\}$ is a J-junta (i.e. depends only on at most J of its n arguments) with query complexity polynomial in J and $1/\epsilon$.

Other research in the area includes the work of Kearns and Ron [10], who gave testing algorithms for the classes of interval functions over the continuous interval $[0, 1]$ and for neural networks and decision trees over the continuous cube $[0, 1]^n$. Their results are not comparable to ours because they differ from the "standard" property testing results in several ways; for one thing, they view the dimension n as a constant and their algorithms have query complexity that depends (exponentially) on n.

Our Results. Our main result is a general algorithm that can be used to test whether an unknown function $f : \Omega^n \to X$ belongs to one of many different representation classes, as long as the representation class satisfies certain conditions. We show that this algorithm yields property testers for many classes that were not previously known to be testable. These include decision lists, size-s decision trees, size-s branching programs, s-term DNF (resolving the aforementioned open question of Parnas *et al.*), size-s Boolean formulas, size-s Boolean circuits, and s-sparse polynomials over $\mathbb{F}_2$.[1] For each of these classes the testing algorithm uses $\text{poly}(s, 1/\epsilon)$ many queries, independent of the number n of inputs to the function (the running time is exponential in s, though linear in n). These testing results are summarized in the top part of Table 1. We note that our general algorithm can also be easily shown to yield property testers for all of the classes tested in [12]; the query complexities would be slightly larger than in [12], but would not require a specialized algorithm for each problem.

Our second contribution is a generalization of the notion of *variation* given in [6] to functions with non-Boolean ranges. This generalization, and the properties we establish for the generalized variation, lets us extend the junta test of [6] to functions with non-Boolean ranges. It also allows us to use our general algorithm to achieve testers for non-Boolean valued function classes such as size-s algebraic circuits, size-s algebraic computation trees, and s-sparse polynomials over finite fields (see the bottom of Table 1).

Our third main contribution is a lower bound; we show that any non-adaptive algorithm to test s-sparse polynomials over finite fields of constant size must make $\tilde{\Omega}(\sqrt{s})$ queries. Since this is within a polynomial factor of our upper bound, this result shows that in at least one instance our general algorithm yields a tester that is nearly optimal. (For testing other representation classes, there is a larger gap between our upper and lower bounds. We give some simple but fairly weak lower bounds for other representation classes in the full version of the paper.)

Our techniques. Our approach combines ideas from the junta test of Fischer *et al.* [6] with ideas from learning theory. The basic idea of using a learning algorithm to do property testing goes back to Goldreich *et al.* [7]. They observed that any proper learning algorithm for a class $\mathcal{C}$ can immediately be used as a testing algorithm for $\mathcal{C}$. (If f belongs to $\mathcal{C}$, then a proper learning algorithm can be used to find a function $f' \in \mathcal{C}$ that f is $\epsilon/2$-close to, while if f is ϵ-far from $\mathcal{C}$ then clearly the proper learning algorithm cannot find any function $f' \in \mathcal{C}$ that f is even ϵ-close to.) However, it is well known that proper learning algorithms for virtually every interesting class of n-variable functions (such as all the classes listed in Table 1, including such simple classes as Boolean literals) must make at least $\Omega(\log n)$ queries. Thus this testing-by-learning approach did not previously yield any strong results for testing interesting function classes.

We get around this impediment by making the key observation that many interesting classes $\mathcal{C}$ of functions are "well-approximated" by juntas in the following sense: every function in $\mathcal{C}$ is close to some function in $\mathcal{C}_J$, where $\mathcal{C}_J \subseteq \mathcal{C}$ and every function in $\mathcal{C}_J$ is a J-junta. For example, every s-term DNF over $\{0, 1\}^n$ is τ-close to an s-term DNF that depends on only $s \log s/\tau$ variables, since each term with more than $\log s/\tau$ variables can be removed from the DNF at the cost of at most τ/s error. Roughly speaking, our algorithm for testing whether f belongs to $\mathcal{C}$ works by attempting to learn the "structure" of the junta in $\mathcal{C}_J$ that f is close to *without actually identifying the relevant variables on which the junta depends*. If the algorithm finds such a junta function, it accepts, and if it does not, it rejects. Our approach can be characterized as *testing by implicit learning* (as opposed to the explicit proper learning in the approach of Goldreich *et al.* [7]), since we are "learning" the structure of the junta to which f is close without explicitly identifying its relevant variables. Indeed, avoiding identifying the relevant variables is what makes it possible to have query complexity independent of n.

[1] We remind the reader that if $\mathcal{C}$ is a subclass of $\mathcal{C}'$, the existence of a testing algorithm for $\mathcal{C}'$ does *not* imply the existence of a testing algorithm for $\mathcal{C}$; thus, for example, our testing result for Boolean circuits does not imply the results for weaker representations such as Boolean formulas or DNF formulas.

Class of functions	Number of Queries	Reference
Boolean functions $f : \{0,1\}^n \to \{0,1\}$		
Boolean literals (dictators), conjunctions	$O(1/\epsilon)$	[12]
s-term monotone DNFs	$\tilde{O}(s^2/\epsilon)$	[12]
J-juntas	$\tilde{O}(J^2/\epsilon)$, $\Omega(J)$ (adaptive)	[6], [3]
decision lists	$\tilde{O}(1/\epsilon^2)$	this paper
size-s decision trees, size-s branching programs, s-term DNFs, size-s Boolean formulas	$\tilde{O}(s^4/\epsilon^2)$, $\Omega(\log s/\log\log s)$ (adaptive)	this paper
s-sparse polynomials over $\mathbb{F}_2$	$\tilde{O}(s^4/\epsilon^2)$, $\tilde{\Omega}(\sqrt{s})$	this paper
size-s Boolean circuits	$\tilde{O}(s^6/\epsilon^2)$	this paper
functions with Fourier degree $\leq d$	$\tilde{O}(2^{6d}/\epsilon^2)$, $\tilde{\Omega}(\sqrt{d})$	this paper
General functions $f : \Omega^n \to X$		
J-juntas	$\tilde{O}(J^2/\epsilon)$	this paper
s-sparse polynomials over field of size $\lvert\mathbb{F}\rvert$	$\tilde{O}((s\lvert\mathbb{F}\rvert)^4/\epsilon^2)$, $\tilde{\Omega}(\sqrt{s})$ for $\lvert\mathbb{F}\rvert = O(1)$	this paper
size-s algebraic circuits, size-s algebraic computation trees over $\mathbb{F}$	$\tilde{O}(s^4 \log^3 \lvert\mathbb{F}\rvert/\epsilon^2)$	this paper

Table 1. Selected previous results on testing function classes. Our upper bounds are for adaptive algorithms, though in all cases very similar bounds for non-adaptive algorithms can be achieved. The lower bounds are for non-adaptive algorithms unless otherwise indicated by (adaptive).

We find the structure of the junta f' in $\mathcal{C}_J$ that f is close to by using the techniques of [6]. As in [6], we begin by randomly partitioning the variables of f into subsets and identifying which subsets contain an influential variable (the random partitioning ensures that with high probability, each subset contains at most one such variable if f is indeed in $\mathcal{C}$). Next, we create a sample of random labeled examples $(x^1, y^1), (x^2, y^2), ..., (x^m, y^m)$, where each x^i is a string of length J (not length n; this is crucial to the query complexity of the algorithm) whose bits correspond to the influential variables of f, and where y^i corresponds with high probability to the value of junta f' on x^i. Finally, we exhaustively check whether any function in $\mathcal{C}_J$ – over J input variables – is consistent with this labeled sample. This step takes at least $|\mathcal{C}_J|$ time steps, which is exponential in s for the classes in Table 1; but since $|\mathcal{C}_J|$ is independent of n we are able to get away with an overall query complexity that is independent of n. (The overall time complexity is linear as a function of n; note that such a runtime dependence on n is inevitable since it takes n time steps simply to prepare a length-n query string to the black-box function.) We explain our testing algorithm in more detail in Section 3.

In order to extend our testing results and the junta testing results in [6] to functions with non-Boolean ranges, we extend the technical definition of *variation* given in [6] to more general functions (intuitively, the variation is a measure of the ability of a set of variables to sway a function's output). We show that this extended definition has the necessary properties to carry the analysis of the junta tests and our test over to this more general setting. We present and analyze our extended definition of variation in Section 3.3.

Finally, we prove our lower bound for testing s-sparse polynomials over finite fields in two stages. We first show that any non-adaptive algorithm that can successfully distinguish a linear form $x_{i_1} + \cdots + x_{i_s}$ (over s randomly selected variables from $x_1, \ldots, x_n$) from a linear form $x_{i_1} + \cdots + x_{i_{s+p}}$ (over $s + p$ randomly selected variables, where p is the characteristic of the finite field) must make $\tilde{\Omega}(\sqrt{s})$ queries. This is a technical generalization of a similar result for $\mathbb{F}_2$ in [6]; the heart of our proof is an extension of a convergence type result about random walks over $\mathbb{Z}_2^q$ with arbitrary step distribution to random walks over $\mathbb{Z}_p^q$. (As an interesting side product, the latter also partially answers a question posed in [6] as to what groups possess a similar convergence type property.) We then prove that every s-sparse polynomial g over finite field $\mathbb{F}$ is "far" from every affine function with at least $s + 1$ non-zero coefficients. This result does not have an analogue in [6] (that paper establishes a lower bound on distinguishing size-s parities from size-$(s + 2)$ parities, and it is trivially true that ev-

ery size-s parity is far from every size-$(s+2)$ parity) and its proof requires several ideas; our argument uses random restrictions chosen according to a distribution that depends on the structure of the polynomial g. We present these results in Section 4.

2. Preliminaries

For $i \in \mathbb{N}$, we denote $[i] \stackrel{\text{def}}{=} \{1, 2, \ldots, i\}$. Throughout the paper, Ω denotes an arbitrary finite set and X denotes an arbitrary finite range set. We will be interested in functions f that map from Ω^n to X. In keeping with the notation of Fischer *et al.* [6] we sometimes write $\mathcal{P}([n])$ to denote the domain Ω^n, and we write $x = (x_1, \ldots, x_n)$ to denote an element of the domain $\mathcal{P}([n])$. An important special case for many of the applications of our main result is when f is a Boolean function over the Boolean hypercube, i.e. $\Omega = \{0,1\}^n$ and $X = \{-1, 1\}$.

We view the domain $\mathcal{P}([n])$ as endowed with the uniform probability measure. Two functions $f_1, f_2 : \mathcal{P}([n]) \to X$ are said to be *ϵ-close* if $\Pr[f_1(x) \neq f_2(x)] \leq \epsilon$, and are *$\epsilon$-far* if $\Pr[f_1(x) \neq f_2(x)] > \epsilon$. We write $\mathbb{E}$ to denote expectation and $\mathbb{V}$ to denote variance.

Let $f : \mathcal{P}([n]) \to X$ be a function and let $I \subseteq [n]$ be a subset of the input coordinates. We define $\mathcal{P}(I)$ to be the set of all partial assignments to the input coordinates x_i for $i \in I$. Thus $\mathcal{P}([n]) = \Omega^n$ is the entire domain of all input vectors of length n. For $w \in \mathcal{P}([n] \setminus I)$ and $z \in \mathcal{P}(I)$, we write $w \sqcup z$ to denote the assignment whose i-th coordinate is w_i if $i \in [n] \setminus I$ and is z_i if $i \in I$.

A function $f : \mathcal{P}([n]) \to X$ is said to be a *J-junta* if there exists a set $\mathcal{J} \subseteq [n]$ of size at most J such that $f(x) = f(y)$ for every two assignments $x, y \in \mathcal{P}([n])$ that agree on $\mathcal{J}$.

Let S be a finite set and $\mathbb{P}, \mathbb{Q}$ be probability measures on it. The *statistical distance* between $\mathbb{P}$ and $\mathbb{Q}$ is defined by $\|\mathbb{P} - \mathbb{Q}\| \stackrel{\text{def}}{=} \max_{A \subseteq S} |\mathbb{P}(A) - \mathbb{Q}(A)|$.

3. The test and an overview of its analysis

In this section we present our testing algorithm and give an intuitive explanation of how it works. We close this section with a detailed statement of our main theorem, Theorem 4, describing the correctness and query complexity of the algorithm.

3.1. Subclass approximators.

Let $\mathcal{C}$ denote a class of functions from $\mathcal{P}([n])$ to X. We will be interested in classes of functions that can be closely approximated by juntas in the class. We have the following:

Definition 1. *For $\tau > 0$, we say that a subclass $\mathcal{C}(\tau) \subseteq \mathcal{C}$ is a $(\tau, J(\tau))$-*approximator *for $\mathcal{C}$ if*

- *$\mathcal{C}(\tau)$ is closed under permutation of variables, i.e. if $f(x_1, \ldots, x_n) \in \mathcal{C}(\tau)$ then $f(x_{\sigma_1}, \ldots, x_{\sigma_n})$ is also in $\mathcal{C}(\tau)$ for every permutation σ of $[n]$; and*
- *for every function $f \in \mathcal{C}$, there is a function $f' \in \mathcal{C}(\tau)$ such that f' is τ-close to f and f' is a $J(\tau)$-junta.*

Typically for us $\mathcal{C}$ will be a class of functions with size bound s in some particular representation, and $J(\tau)$ will depend on s and τ. (A good running example to keep in mind is $\Omega = \{0,1\}$, $X = \{-1,1\}$, and $\mathcal{C}$ is the class of all functions that have s-term DNF representations. In this case we may take $\mathcal{C}(\tau)$ to be the class of all s-term $\log(s/\tau)$-DNFs, and we have $J(\tau) = s\log(s/\tau)$.) Our techniques will work on function classes $\mathcal{C}$ for which $J(\tau)$ is a slowly growing function of $1/\tau$ such as $\log(1/\tau)$. In Section 3.7 we will consider many different specific instantiations of $\mathcal{C}$ and corresponding choices of $\mathcal{C}(\tau)$.

We write $\mathcal{C}(\tau)_k$ to denote the subclass of $\mathcal{C}(\tau)$ consisting of those functions that depend only on variables in $\{x_1, \ldots, x_k\}$. We may (and will) view functions in $\mathcal{C}(\tau)_k$ as taking k arguments from Ω rather than n.

3.2. The independence test.

An important sub-test that will be used throughout the main test is the independence test from [6].

Independence test: Given a function f, and a set of variables I, choose $w \in_{\text{R}} \mathcal{P}([n] \setminus I)$ and $z_1, z_2 \in_{\text{R}} \mathcal{P}(I)$. Accept if $f(w \sqcup z_1) = f(w \sqcup z_2)$ and reject if $f(w \sqcup z_1) \neq f(w \sqcup z_2)$.

If f is independent of the coordinates in I, the independence test always accepts. On the other hand, intuitively if I contains highly relevant variables that are likely to sway the output of f, the independence test is likely to reject.

3.3. Extended variation and testing juntas with non-Boolean ranges.

Fischer *et al.* [6] defined the notion of the *variation* of a function on a subset of input variables. The variation is a measure of the extent to which the function is sensitive to the values of the variables in the set. Let us recall their definition of variation.

Definition 2. *Let f be a function from $\mathcal{P}([n])$ to $\{-1,1\}$, and let $I \subseteq [n]$ be a subset of coordinates. We define the* variation *of f on I as*

$$\mathrm{Vr}_f(I) \stackrel{\text{def}}{=} \mathbb{E}_{w\in\mathcal{P}([n]\setminus I)}\left[\mathbb{V}_{z\in\mathcal{P}(I)}\left[f(w\sqcup z)\right]\right]. \quad (1)$$

Fischer *et al.* showed that the variation is monotone and sub-additive; that for a subset I of the variables, the probability that the independence test rejects is exactly $\frac{1}{2}\mathrm{Vr}_f(I)$; and that if $\mathrm{Vr}_f(I) \leq 2\epsilon$ then f is ϵ-close to a function which does not depend on the variables in I. The analysis of their junta tests depends crucially on these properties of variation.

Unfortunately, the variation properties stated above do not always hold for functions with non-Boolean range, and the original analysis of the junta test does not carry over to the non-Boolean setting. Intuitively, however, the fact that a function may take on more than two values should not make the junta test incorrect. The independence test, which is the main component of the junta test, only checks if values of the function are equal or different. Can one modify the definition of variation and the analysis of the junta test so that the non-Boolean case is captured too?

An approach that we manage to successfully apply is mapping the function range to the Boolean range. The general idea is to pick a mapping from the function range X to the set $\{-1,1\}$ that preserves as much of the sensitivity of the function as possible. If we look at Equation 1 defining variation, we could choose the best mapping to $\{-1,1\}$ either before or after the expectation operator. It turns out that depending on the context, one or the other is more suitable, so we define and use both. Denote by $\mathcal{F}(X)$ the set of all functions from X to $\{-1,1\}$.

Definition 3. *Let f be a function from $\mathcal{P}([n])$ to X, and let $I \subseteq [n]$ be a subset of coordinates. We define the* binary variation *of f on I as*

$$\begin{aligned}\mathrm{BinVr}_f(I) &\stackrel{\text{def}}{=} \max_{g\in\mathcal{F}(X)} \mathrm{Vr}_{g\circ f}(I) \\ &= \max_{g\in\mathcal{F}(X)} \mathbb{E}_{w\in\mathcal{P}([n]\setminus I)}\left[\mathbb{V}_{z\in\mathcal{P}(I)}\left[g(f(w\sqcup z))\right]\right],\end{aligned}$$

and the extreme variation *of f on I as*

$$\mathrm{ExtVr}_f(I) \stackrel{\text{def}}{=} \mathbb{E}_{w\in\mathcal{P}([n]\setminus I)}\left[\max_{g\in\mathcal{F}(X)} \mathbb{V}_{z\in\mathcal{P}(I)}\left[g(f(w\sqcup z))\right]\right].$$

To be able to use both new notions of variation, we need to show that they are related. Probabilistic analysis shows that these two quantities are always within a factor of 4 of each other:

$$\frac{1}{4}\mathrm{ExtVr}_f(I) \leq \mathrm{BinVr}_f(I) \leq \mathrm{ExtVr}_f(I).$$

In the full version of our paper, we prove that the *binary variation* has almost identical properties to the original variation. Namely, we show that the binary variation is also monotone and sub-additive; that the independence test rejects with probability at least $\frac{1}{2}\mathrm{BinVr}_f(I)$; and that if $\mathrm{BinVr}_f(I) \leq \epsilon/4$ for some subset I of the variables of f then f is ϵ-close to a function that does not depend on I. Furthermore, we explain how these properties imply that the three junta tests given by Fischer *et al.* essentially work for functions with non-Boolean ranges as well (with minor modifications). Indeed, the first step of our general testing algorithm $\mathcal{A}$ is essentially the junta test of Fischer *et al.* modified to apply to non-Boolean valued functions. We carefully analyze this first step in the full paper; the results there are easily seen to imply that this first step gives an $\tilde{O}(J^2/\epsilon)$-query junta test for non-Boolean functions as claimed in Table 1.

3.4. Explanation of our testing algorithm.

Our algorithm for testing whether a function $f : \mathcal{P}([n]) \to X$ belongs to $\mathcal{C}$ or is ϵ-far from $\mathcal{C}$ is given in Figures 1 through 3. Given $\epsilon > 0$ and black-box access to f, the algorithm performs three main steps:

1. **Identify critical subsets.** In Step 1, we first randomly partition the variables $x_1, \ldots, x_n$ into r disjoint subsets $I_1, \ldots, I_r$. We then attempt to identify a set of $j \leq J(\tau^\star)$ of these r subsets, which we refer to as *critical* subsets because they each contain a "highly relevant" variable. (For now the value $\tau^\star$ should be thought of as a small quantity; we discuss how this value is selected below.) This step is essentially the same as the 2-sided test for J-juntas from Section 4.2 of Fischer *et al.* [6]. We will show that if f is close to a $J(\tau^\star)$-junta then this step will succeed w.h.p., and if f is far from every $J(\tau^\star)$-junta then this step will fail w.h.p.

2. **Construct a sample.** Let $I_{i_1}, \ldots, I_{i_j}$ be the critical subsets identified in the previous step. In Step 2 we construct a set S of m labeled examples $\{(x^1, y^1), \ldots, (x^m, y^m)\}$, where each x^i is independent and uniformly distributed over $\Omega^{J(\tau^\star)}$. We will show that if f belongs to $\mathcal{C}$, then with high probability there is a fixed $f'' \in \mathcal{C}(\tau^\star)_{J(\tau^\star)}$ such that each y^i is equal to $f''(x^i)$. On the other hand, if f is far from $\mathcal{C}$, then we will show that w.h.p. no such $f'' \in \mathcal{C}(\tau^\star)_{J(\tau^\star)}$ exists.

   To construct each labeled example, we again borrow a technique outlined in [6]. We start with a uniformly random $z \in \Omega^n$. We then attempt to determine how the j highly relevant coordinates of z are

Identify-Critical-Subsets (input is black-box access to $f : \Omega^n \to X$ and $\epsilon > 0$)

1. Partition the variables $x_1, \ldots, x_n$ into r random subsets by assigning each of $x_1, \ldots, x_n$ equiprobably to one of $I_1, \ldots, I_r$.
2. Choose s random subsets $\Lambda_1, \ldots, \Lambda_s \subseteq [r]$ of size $J(\tau^\star)$ by uniformly choosing without repetitions $J(\tau^\star)$ members of $[r]$. Each set Λ_i determines a block $B_i \stackrel{\text{def}}{=} \bigcup_{j \in \Lambda_i} I_j$. (Note that we do not guarantee that the blocks are disjoint.)
3. Apply h iterations of the *independence test* (see Section 3.2) to each block B_i. If all of the independence test iterations applied to block B_i accept, then B_i is declared to be *variation-free*, and all the subsets I_j with $j \in \Lambda_i$ are declared to be variation-free on its behalf.
4. If:
   (a) at least half of the blocks $B_1, \ldots, B_s$ are variation-free; and
   (b) except for at most $J(\tau^\star)$ subsets, every subset in the partition $I_1, \ldots, I_r$ is declared variation-free on behalf of some block,

   then output the list $I_{i_1}, \ldots, I_{i_j}$ of those subsets that are *not* declared to be variation-free. (We call these the *critical* subsets.) Otherwise, halt and output "Not in $\mathcal{C}$."

Figure 1. The subroutine Identify-Critical-Subsets.

set. Although we don't know which of the coordinates of z are highly relevant, we do know that, assuming the previous step was successful, there should be one highly relevant coordinate in each of the critical subsets. We use the independence test repeatedly to determine the setting of the highly relevant coordinate in each critical subset.

For example, suppose that $\Omega = \{0, 1\}$ and I_1 is a critical subset. To determine the setting of the highly relevant coordinate of z in critical subset I_1, we subdivide I_1 into two sets: the subset $\Omega_0 \subseteq I_1$ of indices where z is set to 0, and the subset $\Omega_1 = I_1 \backslash \Omega_0$ of indices where z is set to 1. We can then use the independence test on both Ω_0 and Ω_1 to find out which one contains the highly relevant variable. This tells us whether the highly relevant coordinate of z in subset I_1 is set to 0 or 1. We repeat this process for each critical subset in order to find the settings of the j highly relevant coordinates of z; these form the string x. (The other $J(\tau^\star) - j$ coordinates of x are set to random values; intuitively, this is okay since they are essentially irrelevant.) We then output $(x, f(z))$ as the labeled example.

3. **Check consistency.** Finally, in Step 3 we search through $\mathcal{C}(\tau^\star)_{J(\tau^\star)}$ looking for a function f'' over $\Omega^{J(\tau^\star)}$ that is consistent with all m examples in S. (Note that this step takes $\Omega(|\mathcal{C}(\tau^\star)_{J(\tau^\star)}|)$ time but uses no queries.) If we find such a function then we accept f, otherwise we reject.

3.5. Sketch of the analysis.

We now give an intuitive explanation of the analysis of the test.

Completeness. Suppose f is in $\mathcal{C}$. Then there is some $f' \in \mathcal{C}(\tau^\star)$ that is $\tau^\star$-close to f. Intuitively, $\tau^\star$-close is so close that for the entire execution of the testing algorithm, the black-box function f might as well actually be f' (the algorithm only performs $\ll 1/\tau^\star$ many queries in total, each on a uniform random string, so w.h.p. the view of the algorithm will be the same whether the target is f or f'). Thus, for the rest of this intuitive explanation of completeness, we pretend that the black-box function is f'.

Recall that the function f' is a $J(\tau^\star)$-junta. Let us refer to the variables, x_i, that have $\mathrm{BinVr}_f(x_i) > \theta$ (recall that $\mathrm{BinVr}_f(x_i)$ is a measure of the influence of variable x_i, and θ is some threshold to be defined later) as the *highly relevant* variables of f'. Since f' is a junta, in Step 1 we will be able to identify a collection of $j \leq J(\tau^\star)$ "critical subsets" with high probability. Intuitively, these subsets have the property that:

- each highly relevant variable occurs in one of the critical subsets, and each critical subset contains at most one highly relevant variable (in fact at most one relevant variable for f');
- the variables outside the critical subsets are so "irrelevant" that w.h.p. in all the queries the algorithm makes, it doesn't matter how those variables are

Construct-Sample (input is the list $I_{i_1}, \ldots, I_{i_j}$ output by **Identify-Critical-Subsets** and black-box access to f)

1. Repeat the following m times to construct a set S of m labeled examples $(x, y) \in \Omega^{J(\tau^\star)} \times X$, where $\Omega = \{\omega_0, \omega_1, \ldots, \omega_{|\Omega|-1}\}$:
   (a) Draw z uniformly from Ω^n. Let $X_q \stackrel{\text{def}}{=} \{i : z_i = \omega_q\}$, for each $0 \leq q \leq |\Omega| - 1$.
   (b) For $\ell = 1, \ldots, j$
      i. $w \stackrel{\text{def}}{=} 0$
      ii. For $k = 1, \ldots, \lceil \lg |\Omega| \rceil$
         A. $\Omega_0 \stackrel{\text{def}}{=}$ union of $(X_q \cap I_{i_\ell})$ taken over all $0 \leq q \leq |\Omega| - 1$ such that the k-th bit of q is zero
         B. $\Omega_1 \stackrel{\text{def}}{=}$ union of $(X_q \cap I_{i_\ell})$ taken over all $0 \leq q \leq |\Omega| - 1$ such that the k-th bit of q is one
         C. Apply g iterations of the *independence test* to Ω_0. If any of the g iterations reject, mark Ω_0. Similarly, apply g iterations of the *independence test* to Ω_1; if any of the g iterations reject, mark Ω_1.
         D. If exactly one of Ω_0, Ω_1 (say Ω_b) is marked, set the k-th bit of w to b.
         E. If neither of Ω_0, Ω_1 is marked, set the k-th bit of w to unspecified.
         F. If both Ω_0, Ω_1 are marked, halt and output "no".
      iii. If any bit of w is unspecified, choose w at random from $\{0, 1, \ldots, |\Omega| - 1\}$.
      iv. If $w \notin [0, |\Omega| - 1]$, halt and output "no."
      v. Set $x_\ell = \omega_w$.
   (c) Evaluate f on z, assign the remaining $J(\tau^\star) - j$ coordinates of x randomly, and add the pair $(x, f(z))$ to the sample of labeled examples being constructed.

Figure 2. The subroutine Construct-Sample.

Check-Consistency (input is the sample S output by **Identify-Critical-Subsets**)

1. Check every function in $\mathcal{C}(\tau^\star)_{J(\tau^\star)}$ to see if any of them are consistent with sample S. If so output "yes" and otherwise output "no."

Figure 3. The subroutine Check-Consistency.

set (randomly flipping the values of these variables would not change the value of f' w.h.p.).

Given critical subsets from Step 1 that satisfy the above properties, in Step 2 we construct a sample of labeled examples $S = \{(x^1, y^1), \ldots, (x^m, y^m)\}$ where each x^i is independent and uniform over $\Omega^{J(\tau^\star)}$. We show that w.h.p. there is a $J(\tau^\star)$-junta $f'' \in C(\tau^\star)_{J(\tau^\star)}$ with the following properties:

- there is a permutation $\sigma : [n] \to [n]$ for which $f''(x_{\sigma(1)}, \ldots, x_{\sigma(J(\tau))})$ is close to $f'(x_1, \ldots, x_n)$;
- The sample S is labeled according to f''.

Finally, in Step 3 we do a brute-force search over all of $\mathcal{C}(\tau^\star)_{J(\tau^\star)}$ to see if there is a function consistent with S. Since f'' is such a function, the search will succeed and we output "yes" with high probability overall.

Soundness. Suppose now that f is ϵ-far from $\mathcal{C}$.

One possibility is that f is ϵ-far from every $J(\tau^\star)$-junta; if this is the case then w.h.p. the test will output "no" in Step 1.

The other possibility is that f is ϵ-close to a $J(\tau^\star)$-junta f' (or is itself such a junta). Suppose that this is the case and that the testing algorithm reaches Step 2. In Step 2, the algorithm tries to construct a set of labeled examples that is consistent with f'. The algorithm may fail to construct a sample at all; if this happens then it outputs "no." If the algorithm succeeds in constructing a sample S, then w.h.p. this sample is indeed consistent with f'; but in this case, w.h.p. in Step 3 the algorithm will not find any function $g \in \mathcal{C}(\tau^\star)_{J(\tau^\star)}$ that is consistent with all the examples. (If there were such a function g, then standard arguments in learning theory show that w.h.p. any such function $g \in \mathcal{C}(\tau^\star)_{J(\tau^\star)}$ that is consistent with S is in fact close to f'. Since f' is in turn close to f, this would mean that g is close to f. But g be-

longs to $\mathcal{C}(\tau^\star)_{J(\tau^\star)}$ and hence to $\mathcal{C}$, so this violates the assumption that f is ϵ-far from $\mathcal{C}$.)

3.6. The main theorem.

We now state our main theorem, which is proved in detail in the full paper. The algorithm $\mathcal{A}$ is adaptive, but in the full paper we discuss how to make it non-adaptive with only a slight increase in query complexity.

Theorem 4. *There is an algorithm $\mathcal{A}$ with the following properties:*

Let $\mathcal{C}$ be a class of functions from Ω^n to X. Suppose that for every $\tau > 0$, $\mathcal{C}(\tau) \subseteq \mathcal{C}$ is a $(\tau, J(\tau))$-approximator for $\mathcal{C}$. Suppose moreover that for every $\epsilon > 0$, there is a τ satisfying

$$\tau \leq \kappa\epsilon^2 \cdot [\ln(|\Omega|) \cdot J(\tau)^2 \cdot \ln^2(J(\tau)) \cdot \ln^2(|\mathcal{C}(\tau)_{J(\tau)}|) \cdot \ln\ln(J(\tau)) \cdot \ln(\frac{\ln(|\Omega|)}{\epsilon} \ln|\mathcal{C}(\tau)_{J(\tau)}|)]^{-1}, \quad (2)$$

where $\kappa > 0$ is a fixed absolute constant. Let $\tau^\star$ be the largest value τ satisfying (2) above. Then algorithm $\mathcal{A}$ makes

$$\tilde{O}\left(\frac{\ln|\Omega|}{\epsilon^2} J(\tau^\star)^2 \ln^2(|\mathcal{C}(\tau^\star)_{J(\tau^\star)}|)\right)$$

many black-box queries to f, and satisfies the following:

- *If $f \in \mathcal{C}$ then $\mathcal{A}$ outputs "yes" with probability at least $2/3$;*
- *If f is ϵ-far from $\mathcal{C}$ then $\mathcal{A}$ outputs "no" with probability at least $2/3$.*

Here are some observations to help interpret the bound (2). Note that if $J(\tau)$ grows too rapidly as a function of $1/\tau$, e.g. $J(\tau) = \Omega(1/\sqrt{\tau})$, then there will be no $\tau > 0$ satisfying inequality (2). On the other hand, if $J(\tau)$ grows slowly as a function of $1/\tau$, e.g. $\log(1/\tau)$, then it is may be possible to satisfy (2).

In all of our applications $J(\tau)$ will grow as $O(\log(1/\tau))$, and $\ln|C(\tau)_{J(\tau)}|$ will always be at most $\mathrm{poly}(J(\tau))$, so (2) will always be satisfiable. The most typical case for us will be that $J(\tau) \leq \mathrm{poly}(s)\log(1/\tau)$ (where s is a size parameter for the class of functions in question) and $\ln|C(\tau)_{J(\tau)}| \leq \mathrm{poly}(s) \cdot \mathrm{poly}\log(1/\tau)$, which yields $\tau^\star = \tilde{O}(\epsilon^2)/\mathrm{poly}(s)$ and an overall query bound of $\mathrm{poly}(s)/\tilde{O}(\epsilon^2)$.

3.7. Applications to Boolean and Non-Boolean Functions

Theorem 4 can be used to achieve testing algorithms, in most cases polynomial-query ones, for a wide range of natural and well-studied classes of Boolean functions over the n-dimensional Boolean hypercube (i.e. $\Omega = \{0,1\}$ and $X = \{-1,1\}$), such as s-term DNF. We use Theorem 4 to achieve testing algorithms for several interesting classes of non-Boolean functions as well. These testing results are noted in Table 1; we give detailed statements and proofs of these results in the full paper.

4. Lower bounds for testing sparse polynomials.

One consequence of Theorem 4 is a $\mathrm{poly}(s/\epsilon)$-query algorithm for testing s-sparse polynomials over finite fields of fixed size (independent of n). In this section we present a polynomial lower bound for non-adaptive algorithms for this testing problem. (Detailed proofs for all results in this section are given in the full paper.)

Theorem 5. *Let $\mathbb{F}$ be any fixed finite field, i.e. $|\mathbb{F}| = O(1)$ independent of n. There exists a fixed constant $\epsilon > 0$ (depending on $|\mathbb{F}|$) such that any* non-adaptive *ϵ-testing algorithm for the class of s-sparse polynomials over $\mathbb{F}^n$ must make $\tilde{\Omega}(\sqrt{s})$ queries.*

To prove Theorem 5 we use Yao's principle [16] in (what has become) a standard way for proving lower bounds in property testing (e.g. see [5]). We present two distributions D_{YES} and D_{NO}, the former on inputs satisfying the property (i.e. s-sparse polynomials from $\mathbb{F}^n$ to $\mathbb{F}$), the latter on inputs that are ϵ-far from satisfying it, and show that any deterministic (non-adaptive) algorithm making "few" queries cannot distinguish between a random draw from D_{YES} versus a random draw from D_{NO}. By standard arguments (see for example Lemma 8.1 in [5]), it suffices to argue that for any query set $\mathcal{Q} \subset \mathbb{F}^n$ of cardinality $q = \tilde{O}(\sqrt{s})$ the induced distributions on $\mathbb{F}^q$ (obtained by restricting the randomly chosen functions to these q points) have statistical distance less than $1/3$.

We define both D_{YES} and D_{NO} to be distributions over linear forms from $\mathbb{F}^n$ to $\mathbb{F}$. A random function from D_{YES} is obtained by independently and uniformly (with repetitions) picking s variables from $x_1, \ldots, x_n$ and taking their sum. D_{NO} is defined in the same way, but instead we pick $s+p$ variables, where p is the characteristic of the field $\mathbb{F}$. Clearly, every draw from D_{YES} is an s-sparse polynomial over $\mathbb{F}$, and for $n = \omega((s+p)^2)$, the birthday paradox implies that almost all the probability mass of D_{NO} is on functions with $s+p$ distinct nonzero coefficients. We claim that, for any set of $q = \tilde{O}(\sqrt{s})$ points in $\mathbb{F}^n$, the corresponding induced distributions have statistical distance less than $1/3$.

Let $(G, +)$ be a finite group. A probability measure $\mathbb{P}$ on G induces a random walk on G as follows: Denoting by X_n its position at time n, the walk starts at the identity element and at each step selects an element $\xi_n \in G$ according to $\mathbb{P}$ and goes to $X_{n+1} = \xi_n + X_n$. By arguments parallel to those in Section 6 of [6], the aforementioned claim can be reduced to the following theorem about random walks over $\mathbb{Z}_r^q$:

Theorem 6. *Let r be a prime, $q \in \mathbb{N}^*$ and $\mathbb{P}$ be a probability measure on $\mathbb{Z}_r^q$. Consider the random walk X on $\mathbb{Z}_r^q$ with step distribution $\mathbb{P}$. Let $\mathbb{P}_t$ be the distribution of X at step t. There exists an absolute constant $C > 0$ such that for every $0 < \delta \leq 1/2$, if $t \geq C\frac{\log 1/\delta}{\delta} \cdot r^4 \log r \cdot q^2 \log^2(q+1)$ then $\|\mathbb{P}_t - \mathbb{P}_{t+r}\| \leq \delta$.*

Theorem 6 is a non-trivial generalization of a similar result proved in [6] for the special case $r = 2$. We now give a high-level overview of the overall strategy. Any given $x \in (\mathbb{Z}_r^q)^*$ partitions the space into r non-empty subspaces $V_i^x = \{y \in \mathbb{Z}_r^q : \langle y, x\rangle = i\}$ for $i = 0, 1, \ldots, r-1$. We say that an $x \in (\mathbb{Z}_r^q)^*$ is *degenerate* if there exists some i whose probability measure $\mathbb{P}(V_i^x)$ is "large". We consider two cases: If all the Fourier coefficients of $\mathbb{P}$ are not "very large", then we can show by standard arguments that the walk is close to its stationary (uniform) distribution after the desired number of steps. If, on the other hand, there exists a "very large" Fourier coefficient, then we argue that there must also exist a degenerate direction and we use induction on q.

So far we have shown that any algorithm that can successfully distinguish a random linear form $x_{i_1} + \cdots + x_{i_s}$ from a random linear form $x_{i_1} + \cdots + x_{i_{s+p}}$ must make $\tilde{\Omega}(\sqrt{s})$ queries. To complete the proof of Theorem 5, we must show that every s-sparse polynomial over $\mathbb{F}^n$ is "far" from every linear function of the form $x_{i_1} + \cdots + x_{i_{s+p}}$. We do this via the following new theorem , which may be of independent interest:

Theorem 7. *There exists a function $\epsilon = \epsilon(|\mathbb{F}|)$ such that for any $g : \mathbb{F}^n \to \mathbb{F}$ that is an s-sparse polynomial with $s \leq n-1$, g is ϵ-far from every affine function with at least $s+1$ non-zero coefficients.*

The high-level idea of the proof of Theorem 7 is as follows. Let M be a particular monomial in g, and consider what happens when g is hit with a restriction that fixes all variables that do not occur in M. M itself is not affected by the restriction, but it is possible for a longer monomial to "collapse" onto M and obliterate it (i.e. if M is $x_1x_2^2$ and g contains another monomial $M' = -x_1x_2^2x_3^3$, then a restriction that fixes $x_3 \leftarrow 1$ would cause M' to collapse onto M and in fact obliterate M). We show that g must have a short monomial M (which, however, has degree at least 2) with the following property: for a constant fraction of all possible restrictions of variables not in M, no longer monomial collapses onto M. This implies that for a constant fraction of all such restrictions ρ, the induced polynomial g_ρ is "substantially" different from any affine function (since g_ρ – a polynomial of degree at least two – is not identical to any affine function, it must be "substantially" different since there are only length(M) surviving variables), and hence g itself must be "far" from any affine function.

Lower bounds for other function classes. By adapting techniques of Chockler and Gutfreund [3], we can also obtain $\tilde{\Omega}(\log s)$ lower bounds for many of the other testing problems listed in Table 1. We state and prove these lower bounds in the full version of the paper.

5. Conclusion

Our positive results are all achieved via a single generic algorithm that is not geared toward any particular class of functions. For many classes of interest, the query complexity of this algorithm is $\mathrm{poly}(s, 1/\epsilon)$, but the running time is exponential in s. It would be interesting to study algorithms that are more specifically tailored for classes such as s-term DNF, size-s Boolean formulas, etc. with the aim of obtaining poly(s) runtimes.

One approach to achieving better runtimes is to replace our "implicit learning" step with a more efficient proper learning algorithm (the current learning algorithm simply gathers random examples and exhaustively checks for a consistent hypothesis in the concept class $\mathcal{C}(\tau^\star)_{J(\tau^\star)}$). For some specific concept classes, proper learning is known to be NP-hard, but for other classes, the complexity of proper learning is unknown. The existence of a time-efficient proper learning algorithm for some specific class $\mathcal{C}(\tau^\star)_{J(\tau^\star)}$ would likely yield a time-efficient test in our framework.

Another goal for future work is to strengthen our lower bounds; can poly(s) query lower bounds be obtained for classes such as size-s decision trees, s-term DNF, etc?

References

[1] 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.

[2] 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.

[3] H. Chockler and D. Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, 90(6):301–305, 2004.

[4] P. Diaconis. *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics, Hayward, CA, 1988.

[5] 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.

[6] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. *Journal of Computer & System Sciences*, 68:753–787, 2004.

[7] O. Goldreich, S. Goldwaser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.

[8] C. Jutla, A. Patthak, A. Rudra, and D. Zuckerman. Testing low-degree polynomials over prime fields. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '04)*, pages 423–432, 2004.

[9] T. Kaufman and D. Ron. Testing polynomials over general fields. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '04)*, pages 413–422, 2004.

[10] M. Kearns and D. Ron. Testing problems with sublearning sample complexity. *J. Comp. Sys. Sci.*, 61:428–456, 2000.

[11] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. In *Proceedings of the Twenty-Fourth Annual Symposium on Theory of Computing*, pages 462–467, 1992.

[12] M. Parnas, D. Ron, and A. Samorodnitsky. Testing basic boolean formulae. *SIAM J. Disc. Math.*, 16:20–46, 2002.

[13] D. Štefankovič. Fourier transform in computer science. Master's thesis, University of Chicago, 2000.

[14] A. Terras. *Fourier Analysis on Finite Groups and Applications*. Cambridge University Press, Cambridge, UK, 1999.

[15] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326, 1990.

[16] A. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proceedings of the Seventeenth Annual Symposium on Foundations of Computer Science*, pages 222–227, 1977.